



COORDINATED HIGHWAYS ACTION RESPONSE TEAM  
STATE HIGHWAY ADMINISTRATION

---

# **CHART Release 7 Mapping Release 6 Detailed Design**

**Contract SHA-06-CHART  
Document # WO21-DS-001  
Work Order 21, Deliverable 4**

**March 2, 2011  
By  
CSC**



[illegible]

# Table of Contents

---

<b>1</b>	<b>Introduction.....</b>	<b>1-1</b>
1.1	Purpose .....	1-1
1.2	Objectives .....	1-1
1.3	Scope .....	1-1
1.4	Design Process .....	1-2
1.5	Design Tools .....	1-2
1.6	Work Products .....	1-2
<b>2</b>	<b>Architecture .....</b>	<b>2-1</b>
2.1	Network/Hardware.....	2-1
2.2	Software .....	2-1
2.2.1	COTS Products .....	2-1
2.2.1.1	CHART.....	2-1
2.2.1.2	Mapping .....	2-4
2.2.2	Deployment /Interface Compatibility .....	2-4
2.2.2.1	CHART.....	2-4
2.2.2.2	Mapping .....	2-7
2.3	Security .....	2-7
2.4	Data.....	2-8
2.4.1	Data Storage.....	2-8
2.4.1.1	Database.....	2-8
2.4.1.2	CHART Flat Files .....	2-26
2.4.2	Database Design .....	2-28
2.4.2.1	Integrated Map – Detector Bearing.....	2-28
2.4.2.2	NTCIP Camera Support .....	2-30
2.4.2.3	SCAN Weather Integration .....	2-30
2.4.2.4	Archiving - Changes .....	2-33
<b>3</b>	<b>Key Design Concepts .....</b>	<b>3-1</b>
3.1	Integrated Map – Detector Bearing .....	3-1
3.2	NTCIP Camera Support .....	3-2
3.3	SCAN Weather Integration .....	3-2
3.4	Shift Handoff Report.....	3-3
3.5	Error Processing .....	3-3
3.6	Packaging .....	3-4
3.6.1	CHART.....	3-4
3.6.2	Mapping.....	3-5
3.7	Assumptions and Constraints .....	3-5
<b>4</b>	<b>Use Cases – Integrated Map – Detector Bearing.....</b>	<b>7</b>

<b>4.1 CHART.....</b>	<b>7</b>
4.1.1 R7HighLevel (Use Case Diagram) .....	7
4.1.1.1 Configure TSS (Use Case).....	8
4.1.1.2 Configure Video Sources (Use Case) .....	9
4.1.1.3 Configure Weather Settings (Use Case) .....	9
4.1.1.4 Export TSS Data (Use Case).....	9
4.1.1.5 Manage Camera (Use Case).....	9
4.1.1.6 Manage Traffic Events (Use Case) .....	9
4.1.1.7 Provide Weather Data to Internal Applications (Use Case).....	10
4.1.1.8 Retrieve Weather Data From SCAN (Use Case) .....	10
4.1.1.9 View TSS on Map (Use Case) .....	10
4.1.2 MapAndGISUses (Use Case Diagram).....	11
4.1.2.1 Add Alias (Use Case) .....	11
4.1.2.2 Add Close Devices to Response from Map (Use Case).....	12
4.1.2.3 Create Traffic Event (Use Case) .....	12
4.1.2.4 Edit Alias (Use Case).....	12
4.1.2.5 Edit Traffic Event Properties (Use Case).....	12
4.1.2.6 Get Aliases (Use Case) .....	12
4.1.2.7 Get Counties By State (Use Case) .....	12
4.1.2.8 Get Intersecting Routes (Use Case) .....	12
4.1.2.9 Get Regions By State (Use Case) .....	12
4.1.2.10 Get Routes (Use Case).....	12
4.1.2.11 Get States (Use Case) .....	13
4.1.2.12 Manage Aliases (Use Case) .....	13
4.1.2.13 Navigate Map (Use Case).....	13
4.1.2.14 Perform GIS Query (Use Case) .....	13
4.1.2.15 Remove Alias (Use Case).....	13
4.1.2.16 Select Alias Location (Use Case).....	13
4.1.2.17 Select County (Use Case) .....	13
4.1.2.18 Select Intersecting Feature (Use Case) .....	14
4.1.2.19 Select Intersecting Exit (Use Case).....	14
4.1.2.20 Select Intersecting Milepost (Use Case) .....	14
4.1.2.21 Select Intersecting Route (Use Case).....	14
4.1.2.22 Select Map Layers (Use Case).....	14
4.1.2.23 Select Primary Route (Use Case).....	15
4.1.2.24 Select Region (Use Case).....	15
4.1.2.25 Select State (Use Case) .....	15
4.1.2.26 Select Target Location on Map (Use Case) .....	15
4.1.2.27 Specify Traffic Event Location (Use Case) .....	15
4.1.2.28 Specify Alias Location (Use Case).....	15
4.1.2.29 Specify Device Location (Use Case) .....	15
4.1.2.30 Specify Object Location (Use Case).....	16
4.1.2.31 Use Devices and Traffic Events from Map (Use Case) .....	16
4.1.2.32 View Alias List (Use Case).....	16
4.1.2.33 View Center's Events On Map (Use Case) .....	16
4.1.2.34 View Close Devices on Map (Use Case) .....	16
4.1.2.35 View Devices Close to Traffic Event (Use Case) .....	16
4.1.2.36 View Devices On Map (Use Case) .....	17
4.1.2.37 View Home Page (Use Case).....	17
4.1.2.38 View Home Page Map (Use Case) .....	17
4.1.2.39 View Open Events On Map (Use Case).....	17
4.1.2.40 View Traffic Event Details (Use Case).....	17
4.1.3 MapDeviceAndTrafficEventUses (Use Case Diagram) .....	18
4.1.3.1 Display Camera on Home Monitor (Use Case) .....	18

4.1.3.2	Display Camera on Local Monitor (Use Case) .....	18
4.1.3.3	Edit DMS Response Message (Use Case) .....	18
4.1.3.4	Edit HAR Response Message (Use Case) .....	19
4.1.3.5	Edit Traffic Event Roadway Conditions (Use Case) .....	19
4.1.3.6	Override Control of Camera (Use Case) .....	19
4.1.3.7	Release Control of Camera (Use Case) .....	19
4.1.3.8	Request Control of Camera (Use Case) .....	19
4.1.3.9	Use Camera from Map (Use Case) .....	19
4.1.3.10	Use DMS from Map (Use Case) .....	19
4.1.3.11	Use HAR from Map (Use Case) .....	20
4.1.3.12	Use SHAZAM from Map (Use Case) .....	20
4.1.3.13	Use Traffic Event from Map (Use Case) .....	20
4.1.3.14	Use TSS from Map (Use Case) .....	20
4.1.3.15	View Camera Details Page (Use Case) .....	21
4.1.3.16	View DMS Details Page (Use Case) .....	21
4.1.3.17	View HAR Details Page (Use Case) .....	21
4.1.3.18	View SHAZAM Details Page (Use Case) .....	21
4.1.3.19	View Traffic Event Details Page (Use Case) .....	21
4.1.3.20	View TSS Details Page (Use Case) .....	21
4.1.4	ConfigureTSS (Use Case Diagram) .....	22
4.1.4.1	Edit Map Display Options (Use Case) .....	22
4.1.4.2	Set Bearing (Use Case) .....	22
4.1.4.3	Set Zone Group Display Direction (Use Case) .....	22
4.1.4.4	Set Zone Group Display Order (Use Case) .....	23
<b>4.2</b>	<b>Mapping.....</b>	<b>23</b>
4.2.1	Data Exporter Synchronization .....	23
4.2.1.1	Synchronize Add Events & Devices (Use Case) .....	23
4.2.1.2	Synchronize Update Events & Devices (Use Case) .....	24
4.2.1.3	Synchronize Remove Devices (Use Case) .....	24
4.2.2	DisplayTSS (Use Case Diagram) .....	25
4.2.2.1	Operator (Actor) .....	25
4.2.2.2	View TSSs on Intranet & Internet map (Use Case) .....	25
4.2.2.3	View Zone Group Display on TSS (Use Case) .....	25
<b>5</b>	<b>Detailed Design – Integrated Map – Detector Bearing .....</b>	<b>26</b>
<b>5.1</b>	<b>Human-Machine Interface.....</b>	<b>26</b>
<b>5.2</b>	<b>System Interfaces .....</b>	<b>5-7</b>
5.2.1	Class Diagrams .....	5-7
5.2.1.1	TSSManagement (Class Diagram) .....	5-7
<b>5.3</b>	<b>GUI TSS Data Classes .....</b>	<b>5-16</b>
5.3.1	Class Diagrams .....	5-16
5.3.1.1	GUITSSDataClasses (Class Diagram) .....	5-16
<b>5.4</b>	<b>Package chartlite.servlet.tss .....</b>	<b>5-18</b>
5.4.1	Class Diagrams .....	5-18
5.4.1.1	chartlite.servlet.tss Classes .....	5-18
5.4.2	Sequence Diagrams .....	5-20
5.4.2.1	chartlite.servlet.tss:processUpdateZoneGroupDisplayDirection .....	5-20
5.4.2.2	chartlite.servlet.tss:processUpdateMapDisplayOptions .....	5-22
5.4.2.3	chartlite.servlet.tss:getEditTSSMapDisplayOptionsForm .....	5-24

<b>5.5</b>	<b>Package chartlite.servlet.map .....</b>	<b>5-26</b>
5.5.1	Class Diagrams .....	5-26
5.5.1.1	MapClasses (Class Diagram) .....	5-26
5.5.2	Sequence Diagrams.....	5-28
5.5.2.1	MapReqHdlr:getHomePageMapDataJSON.....	5-28
5.5.2.2	MapReqHdlr:getTrafficEventJSON.....	5-29
5.5.2.3	MapReqHdlr:addJSONFeaturesForTSSLayers .....	5-30
5.5.2.4	MapReqHdlr:getCloseDevicesMapDataJSON .....	5-31
<b>5.6</b>	<b>Package CHART2.TSSManagementModule .....</b>	<b>5-32</b>
5.6.1	Class Diagrams .....	5-32
5.6.1.1	TSSManagementModulePkg .....	5-32
5.6.2	Sequence Diagrams.....	5-40
5.6.2.1	PolledTSSImpl:setMapDisplayOptions .....	5-40
<b>5.7</b>	<b>Mapping Device Editor .....</b>	<b>5-41</b>
5.7.1	Sequence Diagrams.....	5-41
<b>5.8</b>	<b>CHART Intranet &amp; Internet Mapping GUI.....</b>	<b>5-42</b>
<b>5.9</b>	<b>CHART Data Exporter Synchronization (CHART Intranet Map) .....</b>	<b>5-42</b>
5.9.1	Class Diagrams .....	5-42
5.9.1.1	CHARTInventoryHandler Classes.....	5-43
5.9.1.2	CHARTMap.Handlers.CHARTInventoryHandler (Class) .....	5-43
5.9.1.3	CHARTMap.Handlers.DMSInventoryHandler (Class) .....	5-43
5.9.1.4	CHARTMap.Handlers.HARInventoryHandler (Class) .....	5-43
5.9.1.5	CHARTMap.Handlers.SHAZAMInventoryHandler (Class) .....	5-43
5.9.1.6	CHARTMap.Handlers.CHARTEventInventoryHandler (Class) .....	5-43
5.9.1.7	CHARTMap.Handlers.CHARTClosureInventoryHandler (Class) .....	5-43
5.9.1.8	CHARTMap.Handlers.CameraInventoryHandler (Class) .....	5-43
5.9.1.9	CHARTMap.Handlers.TssInventoryHandler (Class) .....	5-44
5.9.2	Sequence Diagram .....	5-44
5.9.2.1	CHART Data Exporter Synchronization .....	5-44
<b>6</b>	<b>Use Cases – NTCIP Camera .....</b>	<b>6-1</b>
<b>6.1</b>	<b>R7 Camera Use Cases.....</b>	<b>6-1</b>
6.1.1	Add Video Source (Use Case) .....	6-2
6.1.2	Administrator (Actor) .....	6-2
6.1.3	Block Flash Video To Public (Use Case) .....	6-2
6.1.4	Choose Camera For Monitor (Use Case) .....	6-2
6.1.5	Choose Monitor For Camera (Use Case) .....	6-2
6.1.6	Configure Video Sources (Use Case) .....	6-2
6.1.7	Configure Flash Streaming Control (Use Case).....	6-2
6.1.8	Configure Multiple Video Sending Devices (Use Case) .....	6-2
6.1.9	Configure Switches (Use Case) .....	6-3
6.1.10	ConfigureEncoders (Use Case).....	6-3
6.1.11	Control Flash Video Streams (Use Case) .....	6-3

6.1.12	Copy Video Source (Use Case) .....	6-3
6.1.13	Display Camera Image (Use Case) .....	6-3
6.1.14	Display Flash Streaming Status (Use Case).....	6-3
6.1.15	Display Multiple Video Sending Devices (Use Case) .....	6-3
6.1.16	Display Video (Use Case).....	6-3
6.1.17	Enable Flash Video to Public (Use Case) .....	6-3
6.1.18	Manage Camera (Use Case).....	6-4
6.1.19	Operator (Actor) .....	6-4
6.1.20	Remove Video Source (Use Case).....	6-4
6.1.21	System (Actor).....	6-4
6.1.22	Update Video Source (Use Case) .....	6-4
6.1.23	View Cameras (Use Case) .....	6-4
<b>6.2</b>	<b>DisplayCamera (Use Case Diagram).....</b>	<b>6-5</b>
6.2.1	Build a Route (Use Case).....	6-5
6.2.2	Camera and Monitor On Different Switch Fabric (Use Case) .....	6-5
6.2.3	Camera and Monitor On Same Switch Fabric (Use Case).....	6-6
6.2.4	Command CoreTec MPEG-4 Decoder (Use Case).....	6-6
6.2.5	Command Decoder (Use Case).....	6-6
6.2.6	Command iMPath MPEG-2 Decoder (Use Case).....	6-6
6.2.7	Command V1500 Switch (Use Case) .....	6-6
6.2.8	Display Camera (Use Case).....	6-6
6.2.9	Display Camera On Monitor (Use Case) .....	6-7
6.2.10	Move To Preset (Use Case) .....	6-7
6.2.11	Override Camera Image Display (Use Case) .....	6-7
6.2.12	StartVideoTour (Use Case).....	6-7
6.2.13	StopVideoTour (Use Case) .....	6-7
6.2.14	Terminate Camera Control (Use Case).....	6-7
<b>6.3</b>	<b>MaintainCamera (Use Case Diagram).....</b>	<b>6-8</b>
6.3.1	Control Camera (Use Case) .....	6-8
6.3.2	Poll Camera (Use Case).....	6-8
6.3.3	Put Camera Online (Use Case) .....	6-9
6.3.4	Request Camera Control (Use Case) .....	6-9
6.3.5	Take Camera Offline (Use Case).....	6-9
6.3.6	Terminate Camera Control (Use Case).....	6-9
6.3.7	View Camera Details for Maint (Use Case) .....	6-9
6.3.8	View Camera List for Maint (Use Case) .....	6-10
6.3.9	View Device Details For Maint (Use Case).....	6-10

6.3.10	View Device List For Maint (Use Case).....	6-10
<b>6.4</b>	<b>ManageCamera (Use Case Diagram).....</b>	<b>6-11</b>
6.4.1	Control Camera (Use Case).....	6-12
6.4.2	Display Camera (Use Case).....	6-12
6.4.3	Display No Video Available Source On Monitor (Use Case) .....	6-12
6.4.4	Manage Camera (Use Case).....	6-12
6.4.5	Manage Camera Control (Use Case) .....	6-12
6.4.6	Move to Preset (Use Case).....	6-12
6.4.7	Poll Camera (Use Case).....	6-12
6.4.8	Put Camera Online (Use Case) .....	6-13
6.4.9	Put Monitor Online (Use Case).....	6-13
6.4.10	Remove Camera From Monitors (Use Case).....	6-13
6.4.11	Revoke Control (Use Case) .....	6-13
6.4.12	Revoke Display (Use Case).....	6-13
6.4.13	Send Camera Commands (Use Case) .....	6-13
6.4.14	Store Presets (Use Case) .....	6-13
6.4.15	Take Camera Offline (Use Case).....	6-14
6.4.16	Take Monitor Offline (Use Case).....	6-14
6.4.17	Terminate Camera Control (Use Case).....	6-14
6.4.18	View Monitor Assignments (Use Case).....	6-14
<b>6.5</b>	<b>ManageCameraControl (Use Case Diagram).....</b>	<b>6-15</b>
6.5.1	Check If Camera Controlled (Use Case).....	6-16
6.5.2	Check If Camera Local Monitor Display (Use Case) .....	6-16
6.5.3	Control Camera (Use Case).....	6-16
6.5.4	Evaluate Camera Control Request (Use Case).....	6-16
6.5.5	Grant Camera Control (Use Case) .....	6-16
6.5.6	Manage Camera Control (Use Case) .....	6-16
6.5.7	Notify Operator of Camera Control Status (Use Case).....	6-17
6.5.8	Override Camera Control (Use Case) .....	6-17
6.5.9	Poll Camera (Use Case).....	6-17
6.5.10	Request Camera Control (Use Case) .....	6-17
6.5.11	Terminate Camera Control (Use Case).....	6-18
<b>6.6</b>	<b>R7VerifyNTCIPCameraCompatibility (Use Case Diagram).....</b>	<b>6-19</b>
6.6.1	Configure NTCIP Camera Compatibility Tester (Use Case).....	6-20
6.6.2	Perform NTCIP Camera Compatibility Tests (Use Case) .....	6-20
6.6.3	Save NTCIP Camera Compatibility Test Results (Use Case).....	6-20
6.6.4	Set Pan-Tilt Speed (Use Case).....	6-20



6.6.5	Test Adjust Focus Camera Command (Use Case) .....	6-20
6.6.6	Test Adjust Iris Camera Command (Use Case) .....	6-20
6.6.7	Test Auto Focus Camera Command (Use Case).....	6-20
6.6.8	Test Auto Iris Camera Command (Use Case).....	6-20
6.6.9	Test Go to Preset Camera Command (Use Case) .....	6-20
6.6.10	Test Pan Camera Command (Use Case) .....	6-21
6.6.11	Test Poll Camera Command (Use Case).....	6-21
6.6.12	Test Power Camera On Off Command (Use Case) .....	6-21
6.6.13	Test Set Default Title Line One Camera Command (Use Case).....	6-21
6.6.14	Test Set Default Title Line Two Camera Command (Use Case) .....	6-21
6.6.15	Test Set Preset Camera Command (Use Case) .....	6-21
6.6.16	Test Tilt Camera Command (Use Case) .....	6-21
6.6.17	Test Zoom Camera Command (Use Case) .....	6-21
<b>6.7</b>	<b>SendCameraCommands (Use Case Diagram).....</b>	<b>6-22</b>
6.7.1	Control Camera (Use Case) .....	6-22
6.7.2	Control COHU 3955 Camera (Use Case) .....	6-23
6.7.3	Control NTCIP Camera (Use Case).....	6-23
6.7.4	Control Surveyor VFT Camera (Use Case) .....	6-23
6.7.5	Execute Command (Use Case) .....	6-23
6.7.6	Execute Command Macro (Use Case) .....	6-23
6.7.7	Execute Simple Command (Use Case) .....	6-24
6.7.8	Process3955ControlRequests (Use Case) .....	6-24
6.7.9	Send Camera Commands (Use Case) .....	6-24
6.7.10	Send to Comm Port (Use Case) .....	6-24
6.7.11	Send to Encoder (Use Case) .....	6-24
6.7.12	SendToCommandProcessor (Use Case) .....	6-24
<b>6.8</b>	<b>View NTCIP Camera Compatibility Test Results (Use Case) .....</b>	<b>6-25</b>
6.8.1	Control Camera (Use Case) .....	6-25
6.8.2	Control COHU 3955 Camera (Use Case) .....	6-25
6.8.3	Control NTCIP Camera (Use Case).....	6-25
6.8.4	Control Surveyor VFT Camera (Use Case) .....	6-26
6.8.5	Execute Command (Use Case) .....	6-26
6.8.6	Execute Command Macro (Use Case) .....	6-26
6.8.7	Execute Simple Command (Use Case) .....	6-26
6.8.8	Operator (Actor) .....	6-26
6.8.9	Process3955ControlRequests (Use Case) .....	6-26
6.8.10	Send Camera Commands (Use Case) .....	6-27

6.8.11	Send to Comm Port (Use Case) .....	6-27
6.8.12	Send to Encoder (Use Case) .....	6-27
6.8.13	SendToCommandProcessor (Use Case) .....	6-27
<b>7</b>	<b>Detailed Design – NTCIP Camera .....</b>	<b>7-28</b>
<b>7.1</b>	<b>Human-Machine Interface.....</b>	<b>7-28</b>
7.1.1	Add/Edit NTCIP Camera .....	7-28
7.1.2	View NTCIP Camera Details.....	7-29
7.1.3	Control NTCIP Camera .....	7-29
7.1.4	NTCIP Camera Compliance Tester .....	7-30
<b>7.2</b>	<b>System Interfaces .....</b>	<b>7-33</b>
7.2.1	Class Diagrams .....	7-33
7.2.1.1	VideoHighLevel (Class Diagram) .....	7-33
7.2.1.2	VideoHighLevel-VideoSource (Class Diagram) .....	7-41
7.2.1.3	VideoControl (Class Diagram) .....	7-48
<b>7.3</b>	<b>Camera Control Module .....</b>	<b>7-55</b>
7.3.1	Class Diagrams .....	7-55
7.3.1.1	CameraControlModule (Class Diagram) .....	7-55
7.3.2	Sequence Diagrams.....	7-67
7.3.2.1	CameraControlModule:AddCamera (Sequence Diagram) .....	7-67
7.3.2.2	CameraControlModule:MoveToPreset (Sequence Diagram) .....	7-68
7.3.2.3	CameraControlModule:SavePreset (Sequence Diagram) .....	7-69
7.3.2.4	CameraControlModule:SetCameraConfiguration (Sequence Diagram) .....	7-70
7.3.2.5	CameraControlModule:TakeCameraOffline (Sequence Diagram) .....	7-71
7.3.2.6	Encoder:receive (Sequence Diagram).....	7-72
7.3.2.7	NTCIPCameraProtocolHdlr:adjPan (Sequence Diagram) .....	7-73
7.3.2.8	NTCIPCameraProtocolHdlr:adjZoom (Sequence Diagram).....	7-74
7.3.2.9	NTCIPCameraProtocolHdlr:calculateControlSpeeds (Sequence Diagram) .....	7-75
7.3.2.10	NTCIPCameraProtocolHdlr:connect (Sequence Diagram).....	7-76
7.3.2.11	NTCIPCameraProtocolHdlr:getZoomPosition (Sequence Diagram) .....	7-77
7.3.2.12	NTCIPCameraProtocolHdlr:moveToPreset (Sequence Diagram) .....	7-78
7.3.2.13	NTCIPCameraProtocolHdlr:poll (Sequence Diagram).....	7-79
7.3.2.14	NTCIPCameraProtocolHdlr:sendMessage (Sequence Diagram).....	7-80
7.3.2.15	NTCIPCameraProtocolHdlr:sendMessageForData (Sequence Diagram).....	7-81
7.3.2.16	NTCIPCameraProtocolHdlr:setLabelText (Sequence Diagram) .....	7-82
7.3.2.17	NTCIPCameraProtocolHdlr:setPresetTitle (Sequence Diagram) .....	7-83
7.3.2.18	NTCIPCameraProtocolhdlr:storePreset (Sequence Diagram) .....	7-84
<b>7.4</b>	<b>NTCIP Camera Compliance Tester .....</b>	<b>7-85</b>
7.4.1	Class Diagrams .....	7-85
7.4.1.1	CameraNTCIPComplianceTesterClasses (Class Diagram) .....	7-85
7.4.2	Sequence Diagrams.....	7-90
7.4.2.1	CameraNTCIPComplianceTester:PanLeft (Sequence Diagram) .....	7-90
<b>7.5</b>	<b>Device Utility .....</b>	<b>7-92</b>
7.5.1	Class Diagrams .....	7-92
7.5.1.1	DeviceUtility (Class Diagram) .....	7-92
7.5.1.2	PortLocatorClasses (Class Diagram) .....	7-97

7.5.2	Sequence Diagrams.....	7-101
7.5.2.1	DataPortUtility:receive (Sequence Diagram) .....	7-101
7.5.2.2	DataPortUtility:receiveFromDirectPort (Sequence Diagram) .....	7-102
7.5.2.3	DataPortUtility:receiveFromTCPPort (Sequence Diagram) .....	7-103
7.5.2.4	DataPortUtility:send (Sequence Diagram).....	7-104
7.5.2.5	NTCIPUtility:get (Sequence Diagram).....	7-105
7.5.2.6	NTCIPUtility:getOEREncodedByteCommand (Sequence Diagram).....	7-106
7.5.2.7	NTCIPUtility:set (Sequence Diagram) .....	7-107

## 8 Use Cases – SCAN Weather Integration .....8-1

### 8.1 R7HighLevel (Use Case Diagram)..... 8-2

8.1.1	Configure TSS (Use Case).....	8-3
8.1.2	Configure Video Sources (Use Case) .....	8-3
8.1.3	Configure Weather Settings (Use Case) .....	8-3
8.1.4	Export TSS Data (Use Case).....	8-3
8.1.5	Manage Camera (Use Case).....	8-4
8.1.6	Manage Traffic Events (Use Case) .....	8-4
8.1.7	Provide Weather Data to Internal Applications (Use Case).....	8-4
8.1.8	Retrieve Weather Data From SCAN (Use Case) .....	8-4
8.1.9	View TSS on Map (Use Case) .....	8-4

### 8.2 ManageTrafficEvents (Use Case Diagram) ..... 8-5

8.2.1	Add Text to Event History (Use Case) .....	8-5
8.2.2	Associate Event (Use Case).....	8-5
8.2.3	Change Event Attributes (Use Case) .....	8-5
8.2.4	Change Event Type (Use Case) .....	8-6
8.2.5	Change Lane Direction (Use Case).....	8-6
8.2.6	Close Event (Use Case) .....	8-6
8.2.7	Copy Traffic Event (Use Case).....	8-6
8.2.8	Create Incident Event (Use Case) .....	8-6
8.2.9	Create Traffic Event (Use Case) .....	8-6
8.2.10	Display Weather Station Conditions (Use Case) .....	8-7
8.2.11	Edit Traffic Event Lane Configuration and Status (Use Case) .....	8-7
8.2.12	Export Priority Event List (Use Case) .....	8-7
8.2.13	Get Event History Text (Use Case) .....	8-7
8.2.14	Log Weather Station Data (Use Case) .....	8-7
8.2.15	Merge Traffic Events (Use Case) .....	8-7
8.2.16	Modify Traffic Event (Use Case) .....	8-8
8.2.17	Operator (Actor) .....	8-8
8.2.18	Preselect Road Surface Condition (Use Case).....	8-8
8.2.19	Query Nearby Weather Station Data (Use Case).....	8-8

8.2.20	Record Lane Closure (Use Case) .....	8-8
8.2.21	Record Organization Notification And Arrival (Use Case) .....	8-9
8.2.22	Record Resource Notification And Arrival (Use Case) .....	8-9
8.2.23	Respond to Traffic Event (Use Case) .....	8-9
8.2.24	Search EORS Permits (Use Case) .....	8-9
8.2.25	Search Traffic Events (Use Case) .....	8-9
8.2.26	Set EORS Permit for Planned Roadway Closure Event (Use Case) .....	8-9
8.2.27	Specify Event Location (Use Case) .....	8-10
8.2.28	Specify Expected Duration (Use Case) .....	8-10
8.2.29	Specify WebsiteTraffic Alert Settings (Use Case) .....	8-10
8.2.30	System (Actor) .....	8-10
8.2.31	Take Event Offline (Use Case) .....	8-10
8.2.32	View Lane Configuration and Status Textually (Use Case) .....	8-11
8.2.33	View Potential Duplicate Events (Use Case) .....	8-11
8.2.34	View Priority Event List (Use Case) .....	8-11
8.2.35	View Suggested EORS Permits (Use Case) .....	8-11
8.2.36	View Traffic Events (Use Case) .....	8-11

## **9 Detailed Design – SCAN Weather Integrations.....9-12**

<b>9.1 Human-Machine Interface.....</b>	<b>9-12</b>
Weather Integration .....	9-12
Traffic Event Details Page .....	9-12
<b>9.2 System Interfaces .....</b>	<b>9-15</b>
9.2.1 Class Diagrams .....	9-15
9.2.1.1 TrafficEventManager2 (Class Diagram) .....	9-15
<b>9.3 Traffic Event Module Package .....</b>	<b>9-21</b>
9.3.1 Class Diagrams .....	9-21
9.3.1.1 TrafficEventModuleClassesR7 (Class Diagram) .....	9-21
9.3.2 Sequence Diagrams.....	9-23
9.3.2.1 TrafficEventFactoryImpl:createTrafficEventHelper (Sequence Diagram) .....	9-23
9.3.2.2 TrafficEventFactoryImpl:getWeatherService (Sequence Diagram) .....	9-25
9.3.2.3 TrafficEventFactoryImpl:queryRoadSurfaceConditionWeatherInfoIfApplicable (Sequence Diagram) .....	9-26
9.3.2.4 TrafficEventGroup:close (Sequence Diagram) .....	9-27
9.3.2.5 TrafficEventGroup:handleRoadSurfaceConditionWeatherInfo (Sequence Diagram) .....	9-28
9.3.2.6 TrafficEventGroup:queryRoadSurfaceWeatherInfo (Sequence Diagram) .....	9-28
<b>9.4 Utility Package .....</b>	<b>9-30</b>
9.4.1 Class Diagrams .....	9-30
9.4.1.1 UtilityClasses3 (Class Diagram) .....	9-30
9.4.2 Sequence Diagrams.....	9-32
9.4.2.1 SynchAsynchQueryUtil:executeTimeLimitedQuery (Sequence Diagram) .....	9-32
<b>9.5 Webservices Weather Module Package .....</b>	<b>9-34</b>

9.5.1	Class Diagrams .....	9-34
9.5.1.1	WeatherModuleClasses (Class Diagram) .....	9-34
9.5.2	Sequence Diagrams.....	9-37
9.5.2.1	WeatherModule:initialize (Sequence Diagram).....	9-37
9.5.2.2	DataRefreshTask:run (Sequence Diagram).....	9-38
9.5.2.3	WeatherDataManager:updateWeatherStationData (Sequence Diagram).....	9-39
9.5.2.4	WeatherDataRequestHandler:handleWeatherDataRequest (Sequence Diagram).....	9-40
9.5.2.5	WeatherDataRequestHandler:calculateRoadSurfaceConditions (Sequence Diagram) .....	9-41
<b>10</b>	<b>Deprecated Functionalities .....</b>	<b>10-1</b>
10.1	CHART Device Editor.....	10-1
10.1.1	View, Add, Update, Remove CHART Devices .....	10-1
<b>11</b>	<b>Mapping To Requirements .....</b>	<b>11-1</b>
<b>12</b>	<b>Acronyms/Glossary .....</b>	<b>12-1</b>

## Table of Figures

Figure 2-1 CHART and External Interfaces .....	2-5
Figure 2-2 R7 Server Deployment.....	2-6
Figure 2-3 R7 GUI Deployment.....	2-7
Figure 2-4 R7 ERD.....	2-22
Figure 4-1. R7HighLevel (Use Case Diagram) .....	8
Figure 4-2. MapAndGISUses (Use Case Diagram).....	11
Figure 4-3. MapDeviceAndTrafficEventUses (Use Case Diagram).....	18
Figure 4-4. ConfigureTSS (Use Case Diagram) .....	22
Figure 4-5. Data Exporter Synchronization (Use Case Diagram).....	23
Figure 4-6 DisplayTSS (Use Case Diagram).....	25
Figure 5-1 The Edit Map Display Options Link in the Zone Groups Configuration Section .....	26
Figure 5-2. The TSS Map Display Options Page. ....	27
Figure 5-3. Setting the TSS Bearing Using the Slider or the Map.....	28
Figure 5-4. Zone Groups Organized by Display Direction. ....	5-1
Figure 5-5. Warning That a Zone Group is Being Changed to Displayable on Maps. ....	5-2
Figure 5-6. Updating the Zone Group Display Order by Using the Move In and Move Out Links. ....	5-3
Figure 5-7. Zone Group Tooltip Showing the Number, Name, and Direction. ....	5-3
Figure 5-8. Map Layer Selector Showing TSS Layers Selected.....	5-4
Figure 5-9. Icons of TSSs in Different States on the Map. ....	5-5
Figure 5-10. TSS Tooltip and Callout Showing Number, Name, Direction, and Zone Groups. ....	5-6
Figure 5-11 TSSManagement (Class Diagram).....	5-8
Figure 5-12 GUITSSDataClasses (Class Diagram).....	5-16
Figure 5-13 chartlite.servlet.tss_classes (Class Diagram) .....	5-18
Figure 5-14 chartlite.servlet.tss:processUpdateZoneGroupDisplayDirection (Sequence Diagram).....	5-21
Figure 5-15 chartlite.servlet.tss:processUpdateMapDisplayOptions (Sequence Diagram) .....	5-23
Figure 5-16 chartlite.servlet.tss:getEditTSSMapDisplayOptionsForm (Sequence Diagram).....	5-25
Figure 5-17 MapClasses (Class Diagram) .....	5-26
Figure 5-18 MapReqHdlr:getHomePageMapDataJSON (Sequence Diagram) .....	5-28
Figure 5-19 :getTrafficEventJSON (Sequence Diagram).....	5-29
Figure 5-20 :addJSONFeaturesForTSSLayers (Sequence Diagram).....	5-30
Figure 5-21 MapReqHdlr:getCloseDevicesMapDataJSON (Sequence Diagram).....	5-31
Figure 5-22 TSSManagementModulePkg (Class Diagram) .....	5-33
Figure 5-23 PolledTSSImpl:setMapDisplayOptions (Sequence Diagram) .....	5-40
Figure 5-24 DataSynchronization: HumanMachine .....	5-41
Figure 5-25 Data Exporter Synchronization (Class Diagram).....	5-42
Figure 5-26 Data Exporter Synchronization (Sequence Diagram) .....	5-45
Figure 6-1. R7CameraUses (Use Case Diagram) .....	6-1
Figure 6-2 Display Camera (Use Case Diagram) .....	6-5
Figure 6-3 MaintainCamera (Use Case Diagram) .....	6-8
Figure 6-4ManageCamera (Use Case Diagram).....	6-11
Figure 6-5 ManageCameraControl (Use Case Diagram).....	6-15
Figure 6-6 R7VerifyNTCIPCameraCompatibility (Use Case Diagram) .....	6-19
Figure 6-7 SendCameraCommands (Use Case Diagram).....	6-22
Figure 6-8 SendCameraCommands (Use Case Diagram).....	6-25
Figure 7-1. NTCIP Camera Compliance Tester (Main Screen).....	7-30
Figure 7-2 NTCIP Camera Compliance Tester (Connecting).....	7-31
Figure 7-3 NTCIP Camera Compliance Tester (Configuration).....	7-32
Figure 7-4 VideoHighLevel (Class Diagram).....	7-34
Figure 7-5 VideoHighLevel-VideoSource (Class Diagram).....	7-42
Figure 7-6 VideoControl (Class Diagram) .....	7-48
Figure 7-7. CameraControlModule (Class Diagram) .....	7-56
Figure 7-8. CameraControlModule:AddCamera (Sequence Diagram).....	7-67
Figure 7-9. CameraControlModule:MoveToPreset (Sequence Diagram) .....	7-68
Figure 7-10. CameraControlModule:SavePreset (Sequence Diagram) .....	7-69

Figure 7-11. CameraControlModule:SetCameraConfiguration (Sequence Diagram) .....	7-70
Figure 7-12. CameraControlModule:TakeCameraOffline (Sequence Diagram) .....	7-71
Figure 7-13. Encoder:receive (Sequence Diagram) .....	7-72
Figure 7-14. NTCIPCameraProtocolHdlr:adjPan (Sequence Diagram) .....	7-73
Figure 7-15. NTCIPCameraProtocolHdlr:adjZoom (Sequence Diagram) .....	7-74
Figure 7-16. NTCIPCameraProtocolHdlr:calculateControlSpeeds (Sequence Diagram) .....	7-75
Figure 7-17. NTCIPCameraProtocolHdlr:connect (Sequence Diagram) .....	7-76
Figure 7-18. NTCIPCameraProtocolHdlr:getZoomPosition (Sequence Diagram) .....	7-77
Figure 7-19. NTCIPCameraProtocolHdlr:moveToPreset (Sequence Diagram) .....	7-78
Figure 7-20. NTCIPCameraProtocolHdlr:poll (Sequence Diagram) .....	7-79
Figure 7-21. NTCIPCameraProtocolHdlr:sendMessage (Sequence Diagram) .....	7-80
Figure 7-22. NTCIPCameraProtocolHdlr:sendMessageForData (Sequence Diagram) .....	7-81
Figure 7-23. NTCIPCameraProtocolHdlr:setLabelText (Sequence Diagram) .....	7-82
Figure 7-24. NTCIPCameraProtocolHdlr:setPresetTitle (Sequence Diagram) .....	7-83
Figure 7-25. NTCIPCameraProtocolHdlr:storePreset (Sequence Diagram) .....	7-84
Figure 7-26. CameraNTCIPComplianceTesterClasses (Class Diagram) .....	7-85
Figure 7-27. CameraNTCIPComplianceTester:PanLeft (Sequence Diagram) .....	7-91
Figure 7-28. DeviceUtility (Class Diagram) .....	7-93
Figure 7-29. PortLocatorClasses (Class Diagram) .....	7-97
Figure 7-30. DataPortUtility:receive (Sequence Diagram) .....	7-101
Figure 7-31. DataPortUtility:receiveFromDirectPort (Sequence Diagram) .....	7-102
Figure 7-32. DataPortUtility:receiveFromTCPPort (Sequence Diagram) .....	7-103
Figure 7-33. DataPortUtility:send (Sequence Diagram) .....	7-104
Figure 7-34. NTCIPUtility:get (Sequence Diagram) .....	7-105
Figure 7-35. NTCIPUtility:getOEREncodedByteCommand (Sequence Diagram) .....	7-106
Figure 7-36. NTCIPUtility:set (Sequence Diagram) .....	7-107
Figure 8-1. R7HighLevel (Use Case Diagram) .....	8-2
Figure 8-2. ManageTrafficEvents (Use Case Diagram) .....	8-5
Figure 9-1. Roadway Conditions Section in Traffic Event .....	9-13
Figure 9-2. Detailed Roadway Conditions .....	9-14
Figure 9-3. TrafficEventManager2 (Class Diagram) .....	9-15
Figure 9-4. TrafficEventModuleClassesR7 (Class Diagram) .....	9-21
Figure 9-5. TrafficEventFactoryImpl:createTrafficEventHelper (Sequence Diagram) .....	9-24
Figure 9-6. TrafficEventFactoryImpl:getWeatherService (Sequence Diagram) .....	9-25
Figure 9-7. TrafficEventFactoryImpl:queryRoadSurfaceConditionWeatherInfoIfApplicable (Sequence Diagram) ..	9-26
Figure 9-8. TrafficEventGroup:close (Sequence Diagram) .....	9-27
Figure 9-9. TrafficEventGroup:handleRoadSurfaceConditionWeatherInfo (Sequence Diagram) .....	9-28
Figure 9-10. TrafficEventGroup:queryRoadSurfaceWeatherInfo (Sequence Diagram) .....	9-29
Figure 9-11. UtilityClasses3 (Class Diagram) .....	9-30
Figure 9-12. SynchAsynchQueryUtil:executeTimeLimitedQuery (Sequence Diagram) .....	9-33
Figure 9-13. WeatherModuleClasses (Class Diagram) .....	9-34
Figure 9-14. WeatherModule:initialize (Sequence Diagram) .....	9-37
Figure 9-15. DataRefreshTask:run (Sequence Diagram) .....	9-38
Figure 9-16. WeatherDataManager:updateWeatherStationData (Sequence Diagram) .....	9-39
Figure 9-17. WeatherDataRequestHandler:handleWeatherDataRequest (Sequence Diagram) .....	9-40
Figure 9-18. WeatherDataRequestHandler:calculateRoadSurfaceConditions (Sequence Diagram) .....	9-42

# 1 Introduction

---

## 1.1 Purpose

This document describes the design of the software for CHART Release 7 and Mapping Release 6. This build provides:

- CHART Map Integration – Detector Bearing and Offset: CHART R7 will move the TSS location editing feature from the Intranet map to CHART and will export that information out of CHART and write it to the existing Intranet map database tables. Both CHART and Mapping are affected by this feature.
- Support for the NTCIP Camera control protocol: CHART R7 will allow NTCIP cameras to be added to the system and to be configured, viewed, and controlled. A separate NTCIP Camera Compliance Tester application is also being developed for use by camera vendors to allow them to determine if their NTCIP cameras can be used in the CHART system. Only CHART is affected by this feature.
- SCAN Integration: CHART R7 will add a weather station feature to CHART. The primary purpose is to automatically populate the Roadway Conditions field within Traffic Events. Only CHART is affected by this feature.
- Shift Handoff Report: CHART R7 will include a Joint Application Design (JAD) session with SHA representatives to determine the basic requirements for an improved Shift Handoff Report. The CSC team will investigate alternative products to replace the existing Wiki that will better allow the requirements of the Shift Handoff Report to be met with minimal or no custom coding required. If a product can be identified, a prototype of the selected product as a Shift Handoff Report will be developed and clarified in three subsequent JAD sessions. Only CHART is affected by this feature.

## 1.2 Objectives

The main objective of this detailed design document is to provide software developers with a framework in which to implement the requirements identified in the CHART R7/Mapping R6 Requirements document. A matrix mapping requirements to the design is presented in Section 15 (Mapping to Requirements).

## 1.3 Scope

This design is limited to Release 7 of the CHART system and Release 6 of the Mapping system, including iMAP. It addresses both the design of the server components of CHART and the Graphical User Interface (GUI) components of CHART to support the new features being added.



Design changes for the Intranet Map and iMAP are included. This design does not include designs for components implemented in earlier releases of the CHART system.

## **1.4 Design Process**

The design was created by capturing the requirements of the system in UML Use Case diagrams. Class diagrams were generated showing the high level objects that address the Use Cases. Sequence diagrams were generated to show how each piece of major functionality will be achieved. This process was iterative in nature – the creation of sequence diagrams sometimes caused re-engineering of the class diagrams, and vice versa.

## **1.5 Design Tools**

The work products contained within this design will be extracted from the Tau Unified Modeling Language (UML) Suite design tool. Within this tool, the design will be contained in the CHART project, Release 7, Analysis phase and System Design phase. And also in the CHART Mapping project, Release 6, Analysis phase and System Design phase.

## **1.6 Work Products**

The final CHART Release 7/Mapping Release 6 design consists of the following work products:

- Use Case diagrams that capture the requirements of the system
- Human-Machine Interface section which provides descriptions of the screens that are changing or being added in order to allow the user to perform the described uses.
- UML Class diagrams, showing the software objects which allow the system to accommodate the uses of the system described in the Use Case diagrams
- UML Sequence diagrams showing how the classes interact to accomplish major functions of the system
- Requirement Verification Traceability Matrix that shows how this design meets the documented requirements for this feature

This document incorporates the four features by providing a Use Cases and Detailed Design section for each feature. For instance, for Integrated Map – Detector Bearing, Use Cases are in Section 4, and Detailed Design (including Human-Machine Interface, Class Diagrams, and Sequence Diagrams) are in Section 5. Sections 6 & 7 cover NTCIP Camera support, and so on.

## 2 Architecture

---

The sections below discuss specific elements of the architecture and software components that are created, changed, or used in CHART Release 7/Mapping Release 6.

### 2.1 Network/Hardware

CHART Release 7/Mapping Release 6 features do not impact the network or hardware architecture of the CHART or Mapping Systems.

### 2.2 Software

CHART uses the Common Object Request Broker Architecture (CORBA) as the base architecture, with custom built software objects made available on the network allowing their data to be accessed via well defined CORBA interfaces. Communications to remote devices use the Field Management Server (FMS) architecture. Newer external interfaces such as the User Management web service, Data Exporter, and GIS service employ a web services architecture combining an HTTP request/response structure to pass XML messages.

Except where noted in the subsections below, CHART Release 7/Mapping Release 6 features do not impact the software architecture of the CHART System.

- For CHART R7, there is a new CHART Weather Web Service that will provide weather related data to internal CHART applications. This Web Service will retrieve data from the SCAN system using an existing / modified interface via the CHARTWeb Database (see database design section).

#### 2.2.1 COTS Products

##### 2.2.1.1 CHART

CHART uses numerous COTS products for both run-time and development. There are no new products being added in Release 7.

The following table contains existing COTS products that have not changed for CHART Release 7:

Product Name	Description
Apache ActiveMQ	CHART uses this to connect to RITIS JMS queues
Apache Jakarta Ant	CHART uses Apache Jakarta Ant 1.6.5 to build CHART applications and deployment jars.
Apache Tomcat	CHART uses Apache Tomcat 6.0.29 as the GUI web server.
Apache XML-RPC	CHART uses the apache xmlrpc java library 3.1.2 protocol that uses XML over HTTP to implement remote procedure calls. The video Flash streaming “red button” (“kill switch”) API uses XML over HTTP remote

Product Name	Description
	procedure calls.
Attention! CC	CHART uses Attention! CC Version 2.1 to provide notification services.
Attention! CC API	CHART uses Attention! CC API Version 2.1 to interface with Attention! CC.
Attention! NS	CHART uses Attention! NS Version 7.0 to provide notification services.
Bison/Flex	CHART uses Bison and Flex as part of the process of compiling binary macro files used for performing camera menu operations on Vicon Surveyor VFT cameras.
bsn.autosuggest	The EORS integration feature uses version 2.1.3 of the bsn.autosuggest JavaScript code from brandspankingnew.net. This tool is freely available and is included as source code in the CHART GUI. It provides a simple JavaScript tool that can be associated with a text entry field. When the user types characters in the field, the tool waits until there has been no typing for a configurable number of milliseconds (to make sure the user is done typing) then places an AJAX call to a web server which can return suggested results that match the user entered text. The bsn.autosuggest tool then parses the results (XML or JSON) and displays a UI element that shows the user the suggestions and lets them select one of them by clicking on it. If a suggested element is selected by the user, a configurable JS method is invoked to allow the application to use the selected suggestion.
CoreTec Decoder Control	CHART uses a CoreTec supplied decoder control API for commanding CoreTec decoders.
Dialogic API	CHART uses the Dialogic API for sending and receiving Dual Tone Multi Frequency (DTMF) tones for HAR communications.
ESRI's ArcGIS Sever	CHART uses version 9.3 to serve maps over the Internet.
ESRI's MapObjects	CHART uses the Map Objects 2.4 for spatial algorithms.
Flex2 SDK	The CHART GUI will use the Flex2 SDK, version 3.1 to provide the Flex compiler, the standard Flex libraries, and examples for building Flex applications.
GIF89 Encoder	Utility classes that can create .gif files with optional animation. This utility is used for the creation of DMS True Display windows.
JAXB	CHART uses the jaxb java library to automate the tedious task of hand-coding field-by-field XML translation and validation for exported data.
JDOM	CHART uses JDOM b7 (beta-7) dated 2001-07-07. JDOM provides a way to represent an XML document for

Product Name	Description
	easy and efficient reading, manipulation, and writing.
JacORB	CHART uses a compiled, patched version of JacORB 2.2.4. The JacORB source code, including the patched code, is kept in the CHART source repository.
Java Run-Time (JRE)	CHART uses 1.6.0_21
JavaService	CHART uses JavaService to install the server side Java software components as Windows services.
JAXEN	CHART uses JAXEN 1.0-beta-8 dated 2002-01-09. The Jaxen project is a Java XPath Engine. Jaxen is a universal object model walker, capable of evaluating XPath expressions across multiple models.
JoeSNMP	CHART uses JoeSNMP version 0.2.6 dated 2001-11-11. JoeSNMP is a Java based implementation of the SNMP protocol. CHART uses for commanding iMPath MPEG-2 decoders and for communications with NTCIP DMSs.
JSON-simple	CHART uses the JSON-simple java library to encode/decode strings that use JSON (JavaScript Object Notation).
JTS	CHART uses the Java Topology Suite (JTS) version 1.8.0 for geographical utility classes.
Log4J	CHART uses the log4J version 1.2.15 for logging purposes.
NSIS	CHART uses the Nullsoft Scriptable Installation System (NSIS), version 2.20, as the server side installation package.
Nuance Text To Speech	For text-to-speech (TTS) conversion CHART uses a TTS engine that integrates with Microsoft Speech Application Programming Interface (MSSAPI), version 5.1. CHART uses Nuance Vocalizer 4.0 with Nuance SAPI 5.1 Integration for Nuance Vocalizer 4.0.
OpenLayers	The Integrated Map feature uses the Open Layers JavaScript API 2.8 ( <a href="http://openlayers.org/">http://openlayers.org/</a> ) in order to render interactive maps within a web application without relying on vendor specific software. Open Layers is an open source product released under a BSD style license which can be found at ( <a href="http://svn.openlayers.org/trunk/openlayers/license.txt">http://svn.openlayers.org/trunk/openlayers/license.txt</a> ).
Oracle	CHART uses Oracle 10.1.0.5 as its database and uses the Oracle 10G JDBC libraries (ojdbc1.4.jar) for all database transactions.
O'Reilly Servlet	Provides classes that allow the CHART GUI to handle

Product Name	Description
	file uploads via multi-part form submission.
Prototype Javascript Library	The CHART GUI uses the Prototype JavaScript library, version 1.6.0.3, a cross-browser compatible JavaScript library provides many features (including easy Ajax support).
SAXPath	CHART uses SAXPath 1.0-beta-6 dated 2001-09-27. SAXPath is an event-based API for XPath parsers, that is, for parsers which parse XPath expressions.
SQLServer JDBC Driver	CHART uses this driver to lookup GIS related data and also to store Location Aliases in SQL Server databases.
Velocity Template Engine	Provides classes that CHART GUI uses in order to create dynamic web pages using velocity templates, CHART uses Velocity version 1.6.1 and tools version 1.4.
Vicon V1500 API	CHART uses a Vicon supplied API for commanding the ViconV1500 CPU to switch video on the Vicon V1500 switch

### 2.2.1.2 Mapping

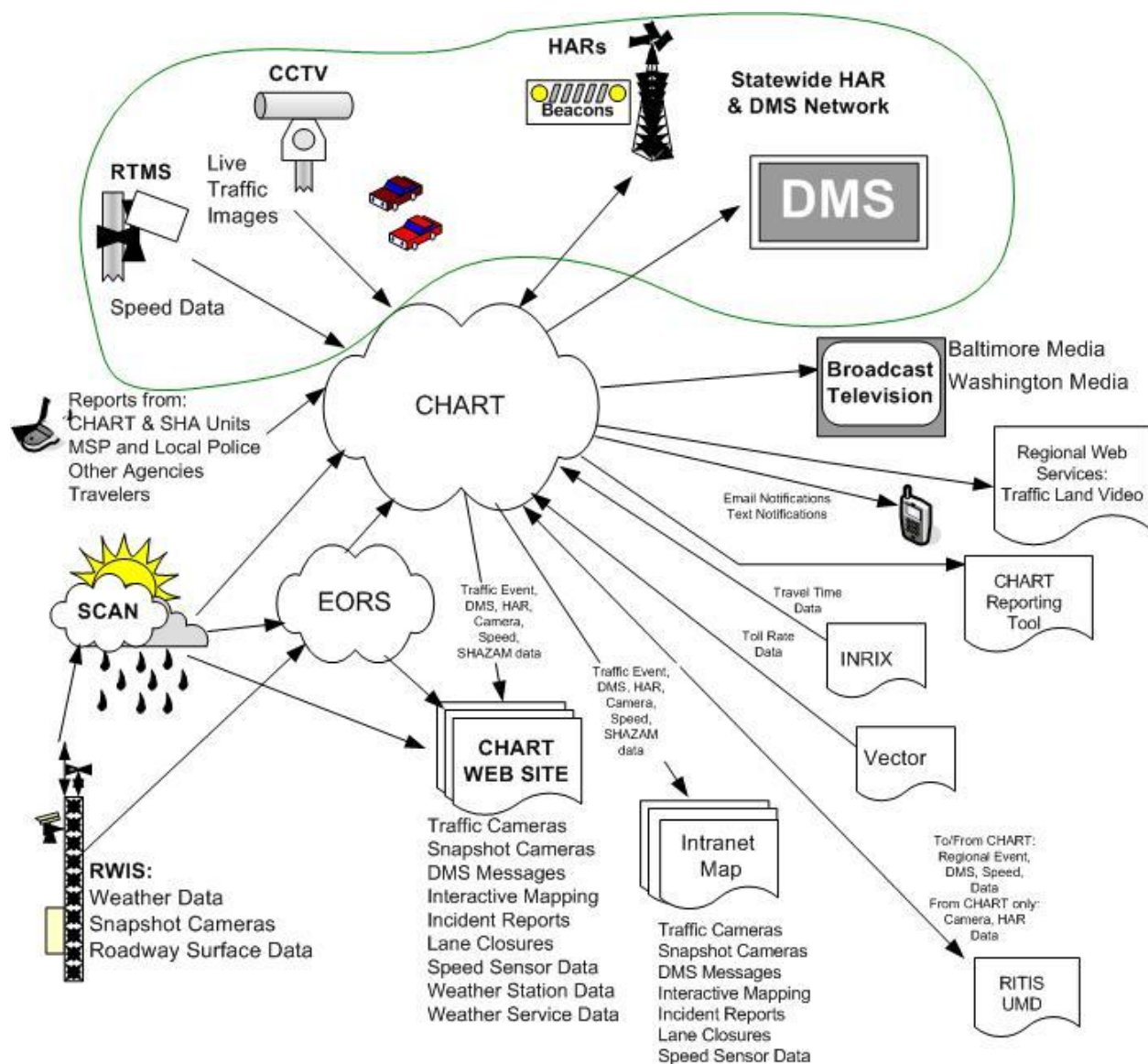
There are no new COTS or COTS upgrades for Mapping Release 6. The ESRI COTS (MapObjects API v 2.4 and ArcSDE v 9.3) will be used. The ArcSDE command line is used to export the original (shape file) spatial table to the CHARTBG (ArcSDE enterprise SDE) database.

## 2.2.2 Deployment /Interface Compatibility

### 2.2.2.1 CHART

#### 2.2.2.1.1 External Interfaces

This section describes the external interfaces being added in Release 7 of the CHART system and Release 6 of the Mapping application.

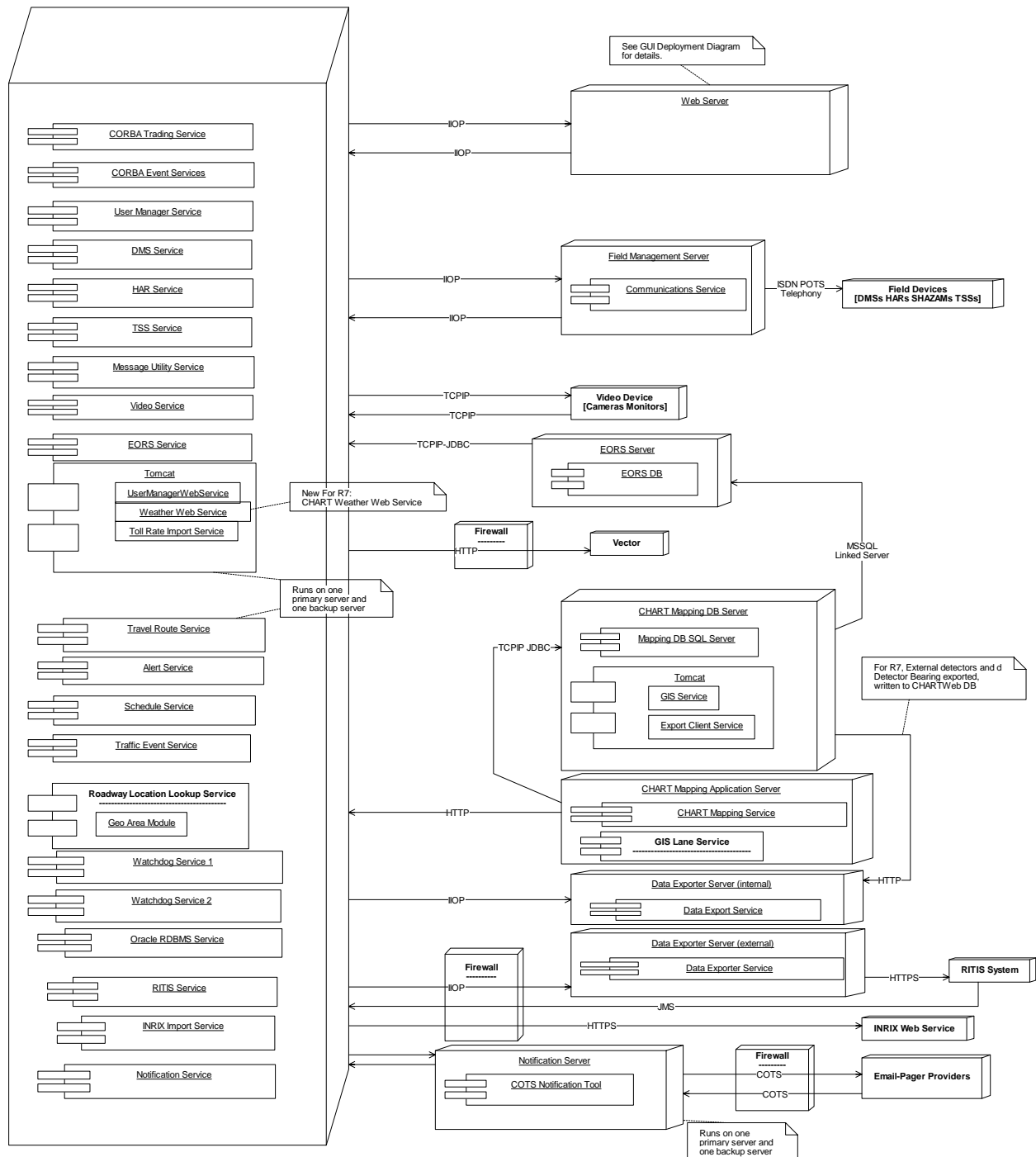


**Figure 2-1 CHART and External Interfaces**

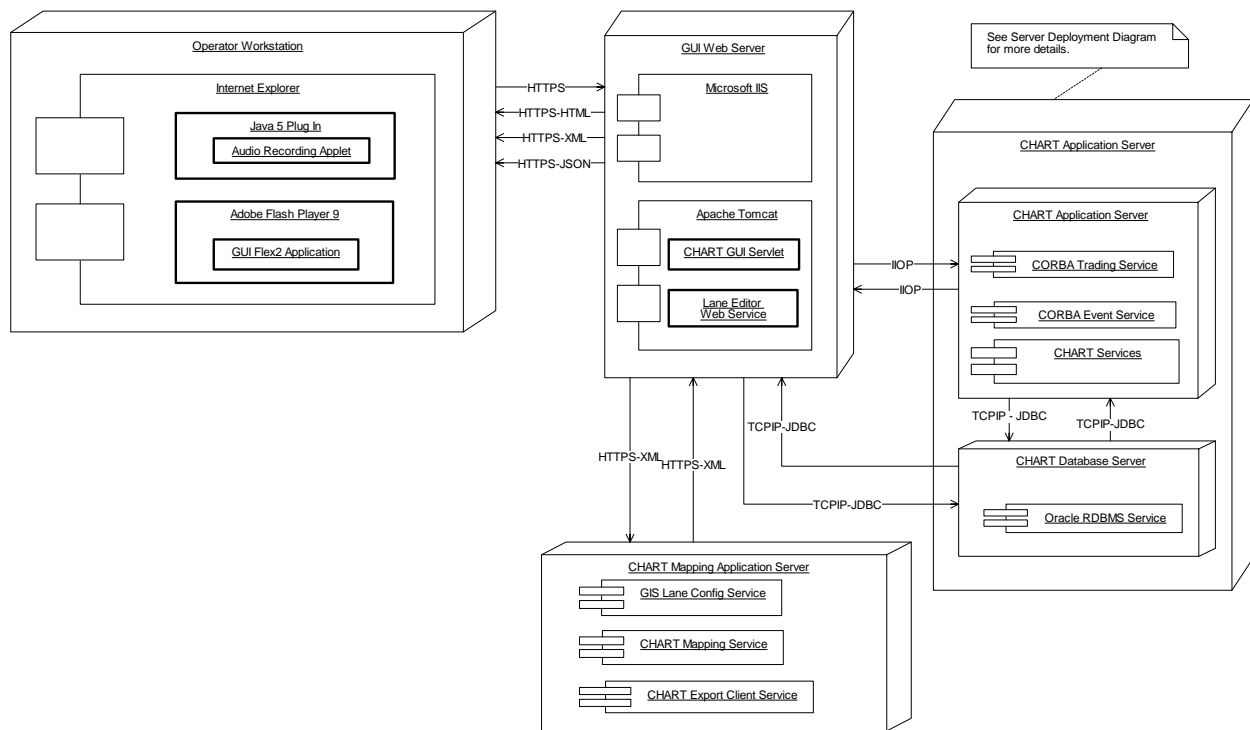
The external interfaces modified/added for R7 are:

1. The R7 Integrated map feature changes the ways that a Detector's bearing and zone groups can be configured. Thus it impacts the CHART Data Exporter and all clients of the CHART Data Exporter by altering the exported XML that describes detector configurations. The CHART Data Exporter Interface Control Document (ICD) has been updated to reflect these changes.
2. For R7, CHAR T will read whether related data from the SCAN databases.

Server and GUI deployment diagrams are shown in the next two figures. The Server Deployment diagram depicts the new weather integration web service and notes changes to the DataExporter and ExportClient for the integrated Map – Detector bearing feature. There are no changes to the GUI deployment for R7.



**Figure 2-2 R7 Server Deployment**



**Figure 2-3 R7 GUI Deployment**

### 2.2.2.1.2 Internal Interfaces

This section describes the internal interfaces being added or modified in Release 7 of the CHART system.

1. The R7 Integrated Map Detector Bearing feature utilizes the existing GUI interface. It changes only the forms that are used by operators to describe detector configuration and map display options. The CHART system IDL has been altered to allow the GUI to pass the new configuration information for the Detector and its constituent zone groups to the TSS Service for persistence and update of other services such as the CHART Data Exporter.
2. The R7 SCAN Weather Integration enhances the GUI by supplying pre-population of a Traffic Event's Road Conditions and the display of additional weather data details as described in the Human Machine Interface section. The CHART system IDL has been altered to support these changes

### 2.2.2.2 Mapping

There are no changes to the existing interfaces for Mapping Release 6.

## 2.3 Security

This section describes the security being added or modified in Release 7 of the CHART system. Unless otherwise noted, features being added for CHART Release 7/Mapping Release 6 do not change security aspects of the CHART or Mapping systems.



## **2.4 Data**

CHART Release 7 will be tested with the currently fielded Oracle database patches. Mapping Release 6 will be tested with the currently fielded SQLServer database patches.

### **2.4.1 Data Storage**

The CHART System stores most of its data in an Oracle database. Additionally the Integrated Map feature adds the ability to store location aliases to the spatial SQL Server database. Some data is stored in flat files on the CHART servers.

The Mapping Application stores and reads its data from a SQLServer database.

This section describes all of these types of data.

#### **2.4.1.1 Database**

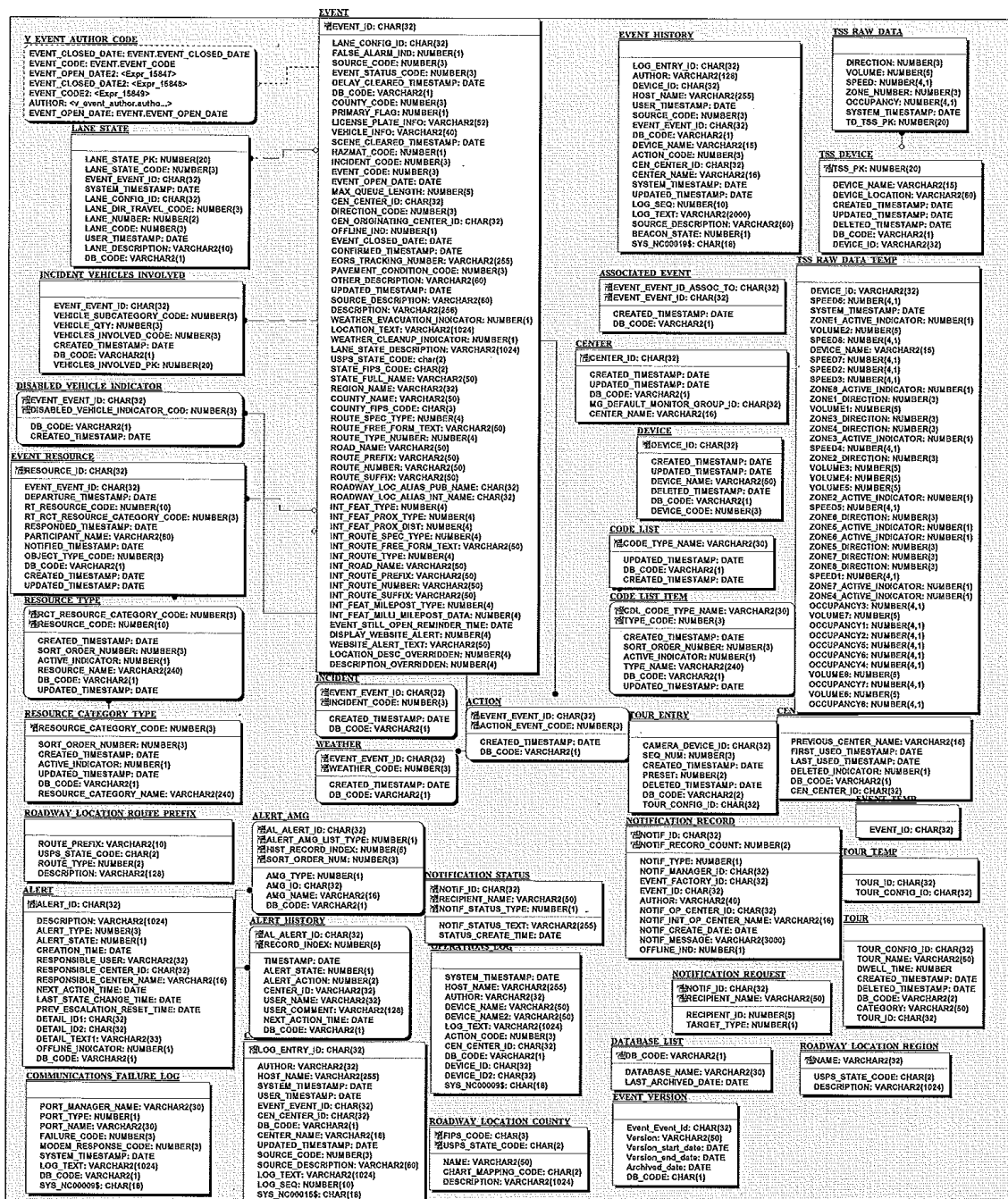
##### **2.4.1.1.1 Database Architecture**

Except as noted CHART Release 7/Mapping Release 6 features do not impact the overall architecture of the CHART database.

##### **2.4.1.1.2 Logical Design**

###### **2.4.1.1.2.1 CHART Entity Relationship Diagram (ERD)**

CHART Database entity relationship diagrams are shown below in the multiple pages of figures labeled collectively as Figure 2-5.



# VIEWS

## VW TOLL RAW DATA

```

TOLL_DATA_IMPORT_ID: TOLL_RAW_DATA.TOLL_DATA_IMPORT_ID
EXT_SYS_START_ID: TOLL_RAW_DATA.EXT_SYS_START_ID
EXT_SYS_END_ID: TOLL_RAW_DATA.EXT_SYS_END_ID
EXT_SYS_ROUTE_DESC: TOLL_RAW_DATA.EXT_SYS_ROUTE_DESC
TOLL_RATE_EXP_TIME: TOLL_RAW_DATA.TOLL_RATE_EXP_TIME
TOLL_RATE_CENTS: TOLL_RAW_DATA.TOLL_RATE_CENTS
DB_CODE: TOLL_RAW_DATA.DB_CODE
ARCHIVED_DATE: TOLL_RAW_DATA.ARCHIVED_DATE
GET_ROWID: TOLL_RAW_DATA.GET_ROWID
HOURS_BEFORE_ARCHIVED_LIVE: TOLL_RAW_DATA.HOURS_BEFORE_ARCHIVED_LIVE
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## VW LINK DATA IMPORT

```

IMPORT_ID: LINK_DATA_IMPORT_IMPORT_ID
SYSTEM_TIMESTAMP: LINK_DATA_IMPORT_IMPORT_ID
EXT_SYS_NAME: LINK_DATA_IMPORT_IMPORT_ID
DB_CODE: LINK_DATA_IMPORT_IMPORT_ID
ARCHIVED_DATE: LINK_DATA_IMPORT_IMPORT_ID
GET_ROWID: LINK_DATA_IMPORT_IMPORT_ID
HOURS_BEFORE_ARCHIVED_LIVE: LINK_DATA_IMPORT_IMPORT_ID
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## VW LINK SMOOTHED DATA

```

LINK_DATA_IMPORT_ID: LINK_SMOOTHED_DATA.LINK_DATA_IMPORT_ID
SYSTEM_TIMESTAMP: LINK_SMOOTHED_DATA.LINK_DATA_IMPORT_ID
EXT_SYS_NAME: LINK_SMOOTHED_DATA.LINK_DATA_IMPORT_ID
LINK_TRAVEL_TIME: LINK_SMOOTHED_DATA.LINK_TRAVEL_TIME
LINK_TRAVEL_TIME_SECS: LINK_SMOOTHED_DATA.LINK_TRAVEL_TIME_SECS
LINK_TRAVEL_TIME_QUAL: LINK_SMOOTHED_DATA.LINK_TRAVEL_TIME_QUAL
LINK_SPEED_MPH: LINK_SMOOTHED_DATA.LINK_SPEED_MPH
DB_CODE: LINK_SMOOTHED_DATA.DB_CODE
ARCHIVED_DATE: LINK_SMOOTHED_DATA.ARCHIVED_DATE
GET_ROWID: LINK_SMOOTHED_DATA.GET_ROWID
HOURS_BEFORE_ARCHIVED_LIVE: LINK_SMOOTHED_DATA.HOURS_BEFORE_ARCHIVED_LIVE
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## VW LINK TRAVEL TIME

```

RL_LINK_ID: LINK_TRAVEL_TIME.LINK_ID
LINK_TRAVEL_TIME: LINK_TRAVEL_TIME.LINK_TRAVEL_TIME
LINK_TRAVEL_TIME_SECS: LINK_TRAVEL_TIME.LINK_TRAVEL_TIME_SECS
LINK_TRAVEL_TIME_QUAL: LINK_TRAVEL_TIME.LINK_TRAVEL_TIME_QUAL
LINK_TRAVEL_TIME_TREND: LINK_TRAVEL_TIME.LINK_TRAVEL_TIME_TREND
DB_CODE: LINK_TRAVEL_TIME.DB_CODE
ARCHIVED_DATE: LINK_TRAVEL_TIME.ARCHIVED_DATE
GET_ROWID: LINK_TRAVEL_TIME.GET_ROWID
HOURS_BEFORE_ARCHIVED_LIVE: LINK_TRAVEL_TIME.HOURS_BEFORE_ARCHIVED_LIVE
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## VW LINK RAW DATA

```

LINK_DATA_IMPORT_ID: LINK_RAW_DATA.LINK_DATA_IMPORT_ID
EXT_SYS_START_ID: LINK_RAW_DATA.EXT_SYS_START_ID
EXT_SYS_END_ID: LINK_RAW_DATA.EXT_SYS_END_ID
EXT_SYS_ROUTE_DESC: LINK_RAW_DATA.EXT_SYS_ROUTE_DESC
TOLL_RATE_EXP_TIME: LINK_RAW_DATA.TOLL_RATE_EXP_TIME
TOLL_RATE_CENTS: LINK_RAW_DATA.TOLL_RATE_CENTS
DB_CODE: LINK_RAW_DATA.DB_CODE
ARCHIVED_DATE: LINK_RAW_DATA.ARCHIVED_DATE
GET_ROWID: LINK_RAW_DATA.GET_ROWID
HOURS_BEFORE_ARCHIVED_LIVE: LINK_RAW_DATA.HOURS_BEFORE_ARCHIVED_LIVE
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## VW ROUTE TOLL RATE

```

TR_ROUTE_ID: ROUTE_TOLL_RATE.ROUTE_ID
TOLL_RATE_EXP_TIME: ROUTE_TOLL_RATE.TOLL_RATE_EXP_TIME
TOLL_RATE_CENTS: ROUTE_TOLL_RATE.TOLL_RATE_CENTS
TOLL_RATE_REASON_CODE: ROUTE_TOLL_RATE.TOLL_RATE_REASON_CODE
TOLL_RATE_INAPPLICABLE_IND: ROUTE_TOLL_RATE.TOLL_RATE_INAPPLICABLE_IND
DB_CODE: ROUTE_TOLL_RATE.DB_CODE
ARCHIVED_DATE: ROUTE_TOLL_RATE.ARCHIVED_DATE
GET_ROWID: ROUTE_TOLL_RATE.GET_ROWID
HOURS_BEFORE_ARCHIVED_LIVE: ROUTE_TOLL_RATE.HOURS_BEFORE_ARCHIVED_LIVE
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## VW ROUTE TRAVEL TIME

```

TR_ROUTE_ID: ROUTE_TRAVEL_TIME.ROUTE_ID
ROUTE_TRAVEL_TIME: ROUTE_TRAVEL_TIME.ROUTE_TRAVEL_TIME
ROUTE_TRAVEL_TIME_SECS: ROUTE_TRAVEL_TIME.ROUTE_TRAVEL_TIME_SECS
ROUTE_TRAVEL_TIME_QUAL: ROUTE_TRAVEL_TIME.ROUTE_TRAVEL_TIME_QUAL
ROUTE_TRAVEL_TIME_TREND: ROUTE_TRAVEL_TIME.ROUTE_TRAVEL_TIME_TREND
DB_CODE: ROUTE_TRAVEL_TIME.DB_CODE
ARCHIVED_DATE: ROUTE_TRAVEL_TIME.ARCHIVED_DATE
GET_ROWID: ROUTE_TRAVEL_TIME.GET_ROWID
HOURS_BEFORE_ARCHIVED_LIVE: ROUTE_TRAVEL_TIME.HOURS_BEFORE_ARCHIVED_LIVE
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## VW ROUTE TRAVEL TIME

```

SYSTEM_TIMESTAMP: DATE
DMS_DEVICE_ID: VARCHAR2(32)
DMS_DEVICE_NAME: VARCHAR2(128)
COMMUNICATION_MODE: NUMBER
OPERATIONAL_STATUS: NUMBER
SCHEDULE_ENABLED_INDICATOR: NUMBER
ENABLED_DMS_TRAV_ROUTE_MSG_ID: VARCHAR2(32)
DMS_MESSAGE: VARCHAR2(1024)
DMS_TRAV_ROUTE_MSG_STATE: NUMBER
DMS_TRAV_ROUTE_MSG_REASON: VARCHAR2(4000)
DB_CODE: CHAR(1)
ARCHIVED_DATE: DATE
HOURS_BEFORE_ARCHIVED_LIVE: NUMBER
STAT_SEQ_SEQUENCE: NUMBER
GET_ROWID: ROWID

```

## TOLL RAW DATA

```

TOLL_DATA_IMPORT_ID: NUMBER
EXT_SYS_START_ID: VARCHAR2(35)
EXT_SYS_END_ID: VARCHAR2(35)
EXT_SYS_ROUTE_DESC: VARCHAR2(127)
TOLL_RATE_EXP_TIME: DATE
TOLL_RATE_CENTS: NUMBER(5)
DB_CODE: CHAR(1)
ARCHIVED_DATE: DATE
HOURS_BEFORE_ARCHIVED_LIVE: NUMBER
GET_ROWID: ROWID

```

## ROUTE TRAVEL TIME

```

TR_ROUTE_ID: CHAR(32)
ROUTE_TRAVEL_TIME_EXP_TIME: DATE
ROUTE_TRAVEL_TIME_SECS: NUMBER(5)
ROUTE_TRAVEL_TIME_TREND: NUMBER(1)
TRAVEL_TIME_INAPPLICABLE_IND: NUMBER(1)
DB_CODE: CHAR(1)
ARCHIVED_DATE: DATE
HOURS_BEFORE_ARCHIVED_LIVE: NUMBER
GET_ROWID: ROWID

```

## ROUTE TRAVEL TIME SECS

```

TR_ROUTE_ID: CHAR(32)
ROUTE_TRAVEL_TIME_EXP_TIME: DATE
ROUTE_TRAVEL_TIME_CALC: VARCHAR2(6000)
ROUTE_TRAVEL_TIME_REASON_CODE: NUMBER(2)
DB_CODE: CHAR(1)
ARCHIVED_DATE: DATE
HOURS_BEFORE_ARCHIVED_LIVE: NUMBER
GET_ROWID: ROWID

```

## DMS TRAV ROUTE MSG LOGGING LOG

```

SYSTEM_TIMESTAMP: DATE
DMS_DEVICE_ID: CHAR(32)
DMS_DEVICE_NAME: VARCHAR2(128)
SCHEDULE_CONFIG_FLAG: NUMBER(2)
DB_CODE: CHAR(1)
ARCHIVED_DATE: DATE
HOURS_BEFORE_ARCHIVED_LIVE: NUMBER
GET_ROWID: ROWID

```

## VW DMS TRAV RT MSG CONFIG LOG

```

SYSTEM_TIMESTAMP: DMS_TRAV_ROUTE_MSG_CONFIG_LOG.SYSTEM_TIMESTAMP
DMS_DEVICE_ID: DMS_TRAV_ROUTE_MSG_CONFIG_LOG.DMS_DEVICE_ID
DMS_DEVICE_NAME: DMS_TRAV_ROUTE_MSG_CONFIG_LOG.DMS_DEVICE_NAME
SCHEDULE_CONFIG_FLAG: DMS_TRAV_ROUTE_MSG_CONFIG_LOG.SCHEDULE_CONFIG_FLAG
DB_CODE: DMS_TRAV_ROUTE_MSG_CONFIG_LOG.DB_CODE
ARCHIVED_DATE: DMS_TRAV_ROUTE_MSG_CONFIG_LOG.ARCHIVED_DATE
GET_ROWID: DMS_TRAV_ROUTE_MSG_CONFIG_LOG.GET_ROWID
HOURS_BEFORE_ARCHIVED_LIVE: DMS_TRAV_ROUTE_MSG_CONFIG_LOG.HOURS_BEFORE_ARCHIVED_LIVE
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## VW DMS TRAV RT MSG ROUTES LOG

```

SYSTEM_TIMESTAMP: DMS_TRAV_ROUTE_MSG_ROUTES_LOG.SYSTEM_TIMESTAMP
DMS_DEVICE_ID: DMS_TRAV_ROUTE_MSG_ROUTES_LOG.DMS_DEVICE_ID
DMS_TRAV_ROUTE_MSG_ID: DMS_TRAV_ROUTE_MSG_ROUTES_LOG.DMS_TRAV_ROUTE_MSG_ID
DMS_TRAV_ROUTE_MSG_REASON: DMS_TRAV_ROUTE_MSG_ROUTES_LOG.DMS_TRAV_ROUTE_MSG_REASON
DB_CODE: DMS_TRAV_ROUTE_MSG_ROUTES_LOG.DB_CODE
ARCHIVED_DATE: DMS_TRAV_ROUTE_MSG_ROUTES_LOG.ARCHIVED_DATE
GET_ROWID: DMS_TRAV_ROUTE_MSG_ROUTES_LOG.GET_ROWID
HOURS_BEFORE_ARCHIVED_LIVE: DMS_TRAV_ROUTE_MSG_ROUTES_LOG.HOURS_BEFORE_ARCHIVED_LIVE
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## VW DMS TRAV RT MSG LOG

```

SYSTEM_TIMESTAMP: DMS_TRAV_ROUTE_MSG_LOG.SYSTEM_TIMESTAMP
MSG_SEQ_SEQUENCE: DMS_TRAV_ROUTE_MSG_LOG.MSG_SEQ_SEQUENCE
DMS_DEVICE_ID: DMS_TRAV_ROUTE_MSG_LOG.DMS_DEVICE_ID
DMS_TRAV_ROUTE_MSG_ID: DMS_TRAV_ROUTE_MSG_LOG.DMS_TRAV_ROUTE_MSG_ID
DMS_TRAV_ROUTE_MSG_REASON: DMS_TRAV_ROUTE_MSG_LOG.DMS_TRAV_ROUTE_MSG_REASON
DB_CODE: DMS_TRAV_ROUTE_MSG_LOG.DB_CODE
ARCHIVED_DATE: DMS_TRAV_ROUTE_MSG_LOG.ARCHIVED_DATE
GET_ROWID: DMS_TRAV_ROUTE_MSG_LOG.GET_ROWID
HOURS_BEFORE_ARCHIVED_LIVE: DMS_TRAV_ROUTE_MSG_LOG.HOURS_BEFORE_ARCHIVED_LIVE
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## VW TRAVEL TIME MSG STATUS LOG

```

SYSTEM_TIMESTAMP: DMS_TRAV_ROUTE_MSG_STATUS_LOG.SYSTEM_TIMESTAMP
MSG_SEQ_SEQUENCE: DMS_TRAV_ROUTE_MSG_STATUS_LOG.MSG_SEQ_SEQUENCE
DMS_DEVICE_ID: DMS_TRAV_ROUTE_MSG_STATUS_LOG.DMS_DEVICE_ID
DMS_TRAV_ROUTE_MSG_ID: DMS_TRAV_ROUTE_MSG_STATUS_LOG.DMS_TRAV_ROUTE_MSG_ID
COMMUNICATION_MODE: DMS_TRAV_ROUTE_MSG_STATUS_LOG.COMMUNICATION_MODE
OPERATIONAL_STATUS: DMS_TRAV_ROUTE_MSG_STATUS_LOG.OPERATIONAL_STATUS
SCHEDULE_ENABLED_INDICATOR: DMS_TRAV_ROUTE_MSG_STATUS_LOG.SCHEDULE_ENABLED_INDICATOR
ENABLED_DMS_TRAV_ROUTE_MSG_ID: DMS_TRAV_ROUTE_MSG_STATUS_LOG.ENABLED_DMS_TRAV_ROUTE_MSG_ID
DMS_MESSAGE: DMS_TRAV_ROUTE_MSG_STATUS_LOG.DMS_MESSAGE
DMS_TRAV_ROUTE_MSG_STATE: DMS_TRAV_ROUTE_MSG_STATUS_LOG.DMS_TRAV_ROUTE_MSG_STATE
DMS_TRAV_ROUTE_MSG_REASON: DMS_TRAV_ROUTE_MSG_STATUS_LOG.DMS_TRAV_ROUTE_MSG_REASON
DB_CODE: DMS_TRAV_ROUTE_MSG_STATUS_LOG.DB_CODE
ARCHIVED_DATE: DMS_TRAV_ROUTE_MSG_STATUS_LOG.ARCHIVED_DATE
GET_ROWID: DMS_TRAV_ROUTE_MSG_STATUS_LOG.GET_ROWID
HOURS_BEFORE_ARCHIVED_LIVE: DMS_TRAV_ROUTE_MSG_STATUS_LOG.HOURS_BEFORE_ARCHIVED_LIVE
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## VW ROUTE TRAVEL TIME TEXT

```

TR_ROUTE_ID: ROUTE_TRAVEL_TIME.TEXT.ROUTE_ID
ROUTE_TRAVEL_TIME_EXP_TIME: ROUTE_TRAVEL_TIME.TEXT.ROUTE_TRAVEL_TIME_EXP_TIME
ROUTE_TRAVEL_TIME_CALC: ROUTE_TRAVEL_TIME.TEXT.ROUTE_TRAVEL_TIME_CALC
ROUTE_TRAVEL_TIME_REASON_CODE: ROUTE_TRAVEL_TIME.TEXT.ROUTE_TRAVEL_TIME_REASON_CODE
DB_CODE: ROUTE_TRAVEL_TIME.TEXT.DB_CODE
ARCHIVED_DATE: ROUTE_TRAVEL_TIME.TEXT.ARCHIVED_DATE
GET_ROWID: ROUTE_TRAVEL_TIME.TEXT.GET_ROWID
HOURS_BEFORE_ARCHIVED_LIVE: ROUTE_TRAVEL_TIME.TEXT.HOURS_BEFORE_ARCHIVED_LIVE
HOURS_AFTER_ARCHIVED: <{Sysdate} - ARCHIVED_DATE>

```

## LINK TRAVEL TIME

```

RL_LINK_ID: CHAR(32)
LINK_TRAVEL_TIME_EXP_TIME: DATE
LINK_TRAVEL_TIME_SECS: NUMBER(5)
LINK_TRAVEL_TIME_QUAL: NUMBER(2)
LINK_TRAVEL_TIME_TREND: NUMBER(1)
DB_CODE: CHAR(1)
ARCHIVED_DATE: DATE
HOURS_BEFORE_ARCHIVED_LIVE: NUMBER
GET_ROWID: ROWID

```

## LINK SMOOTHED DATA

```

LINK_DATA_IMPORT_ID: NUMBER
EXT_LINK_ID: CHAR(9)
LINK_TRAVEL_TIME_EXP_TIME: DATE
LINK_TRAVEL_TIME_SECS: NUMBER(5)
LINK_TRAVEL_TIME_QUAL: NUMBER(2)
LINK_SPEED_MPH: NUMBER(3)
DB_CODE: CHAR(1)
ARCHIVED_DATE: DATE
HOURS_BEFORE_ARCHIVED_LIVE: NUMBER
GET_ROWID: ROWID

```

## LINK RAW DATA

```

LINK_DATA_IMPORT_ID: NUMBER
EXT_LINK_ID: CHAR(9)
LINK_TRAVEL_TIME_EXP_TIME: DATE
LINK_TRAVEL_TIME_SECS: NUMBER(5)
LINK_TRAVEL_TIME_QUAL: NUMBER(2)
LINK_SPEED_MPH: NUMBER(3)
DB_CODE: CHAR(1)
ARCHIVED_DATE: DATE
HOURS_BEFORE_ARCHIVED_LIVE: NUMBER
GET_ROWID: ROWID

```

## ROUTE TOLL RATE

```

TR_ROUTE_ID: CHAR(32)
TOLL_RATE_EXP_TIME: DATE
TOLL_RATE_CENTS: NUMBER(5)
TOLL_RATE_REASON_CODE: NUMBER(2)
TOLL_RATE_INAPPLICABLE_IND: NUMBER(1)
DB_CODE: CHAR(1)
ARCHIVED_DATE: DATE
HOURS_BEFORE_ARCHIVED_LIVE: NUMBER
GET_ROWID: ROWID

```

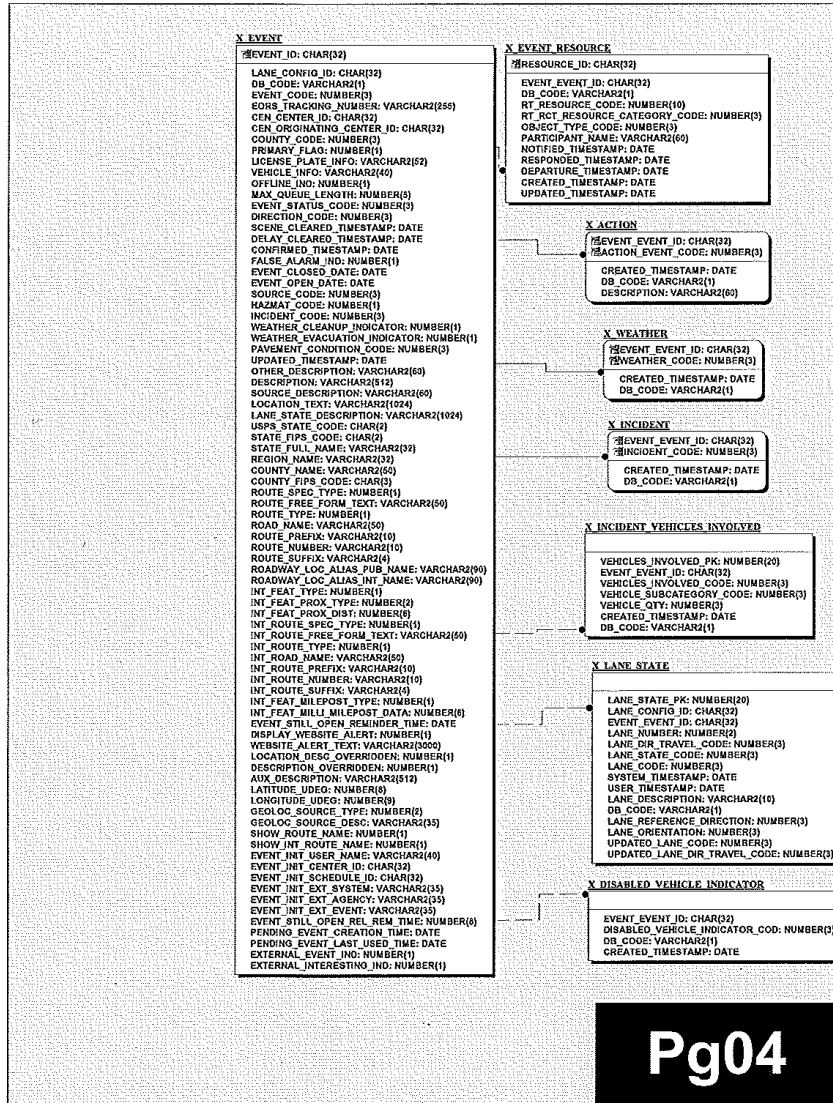
## MS TRAV ROUTE MSG MSGS LOG

```

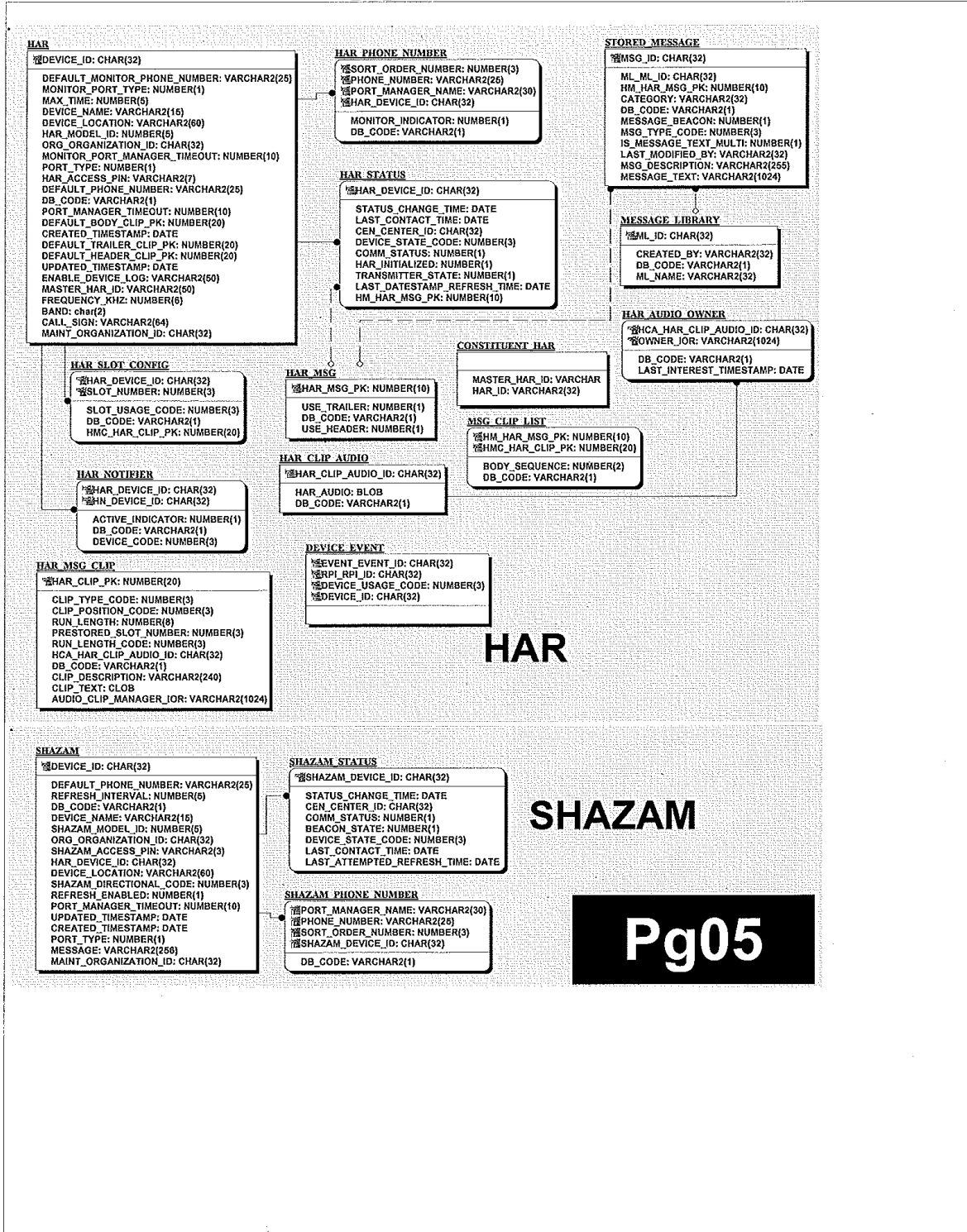
SYSTEM_TIMESTAMP: DATE
MSG_SEQ_SEQUENCE: NUMBER
DMS_DEVICE_ID: VARCHAR2(32)
DMS_TRAV_ROUTE_MSG_ID: VARCHAR2(32)
DMS_TRAV_ROUTE_MSG_TEMPLATE_ID: VARCHAR2(4000)
DB_CODE: CHAR(1)
ARCHIVED_DATE: DATE
HOURS_BEFORE_ARCHIVED_LIVE: NUMBER
GET_ROWID: ROWID

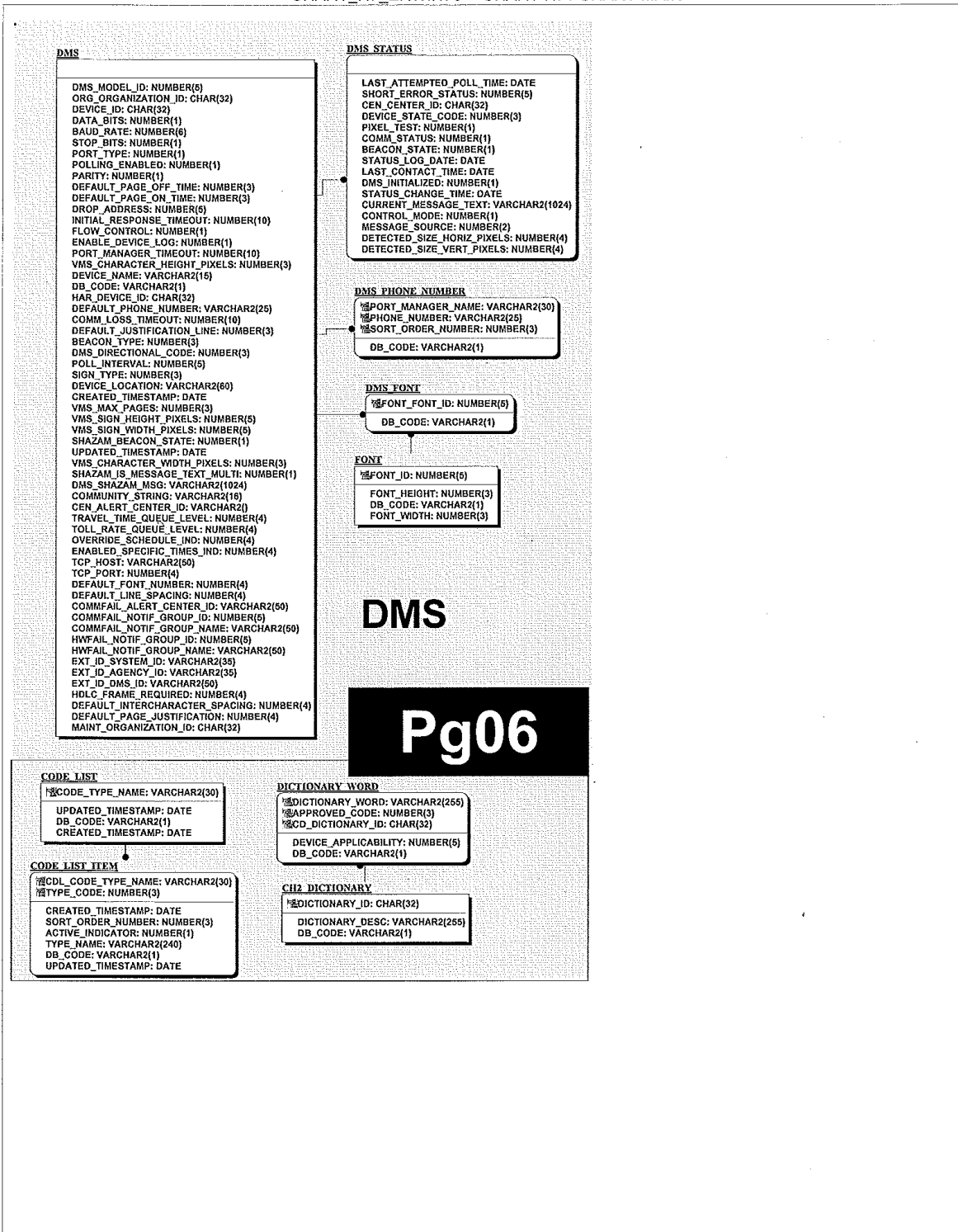
```

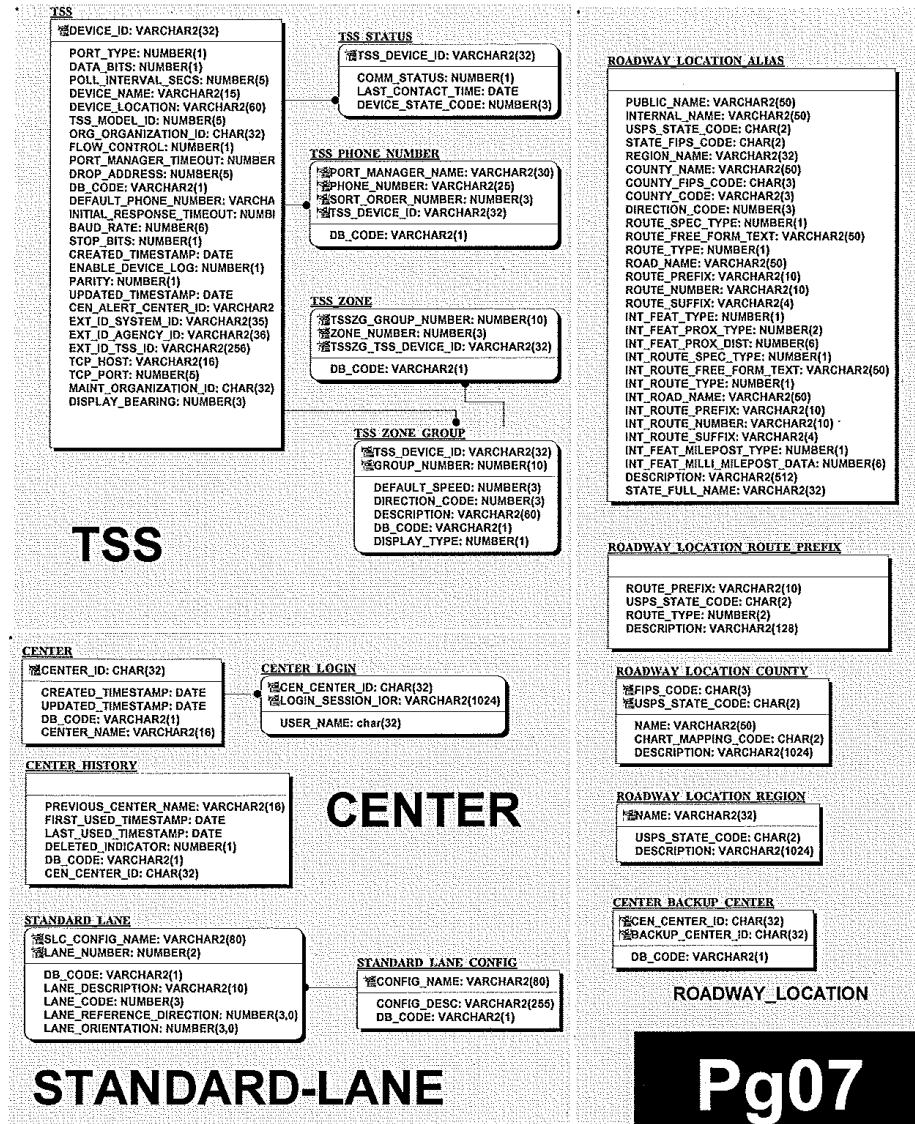




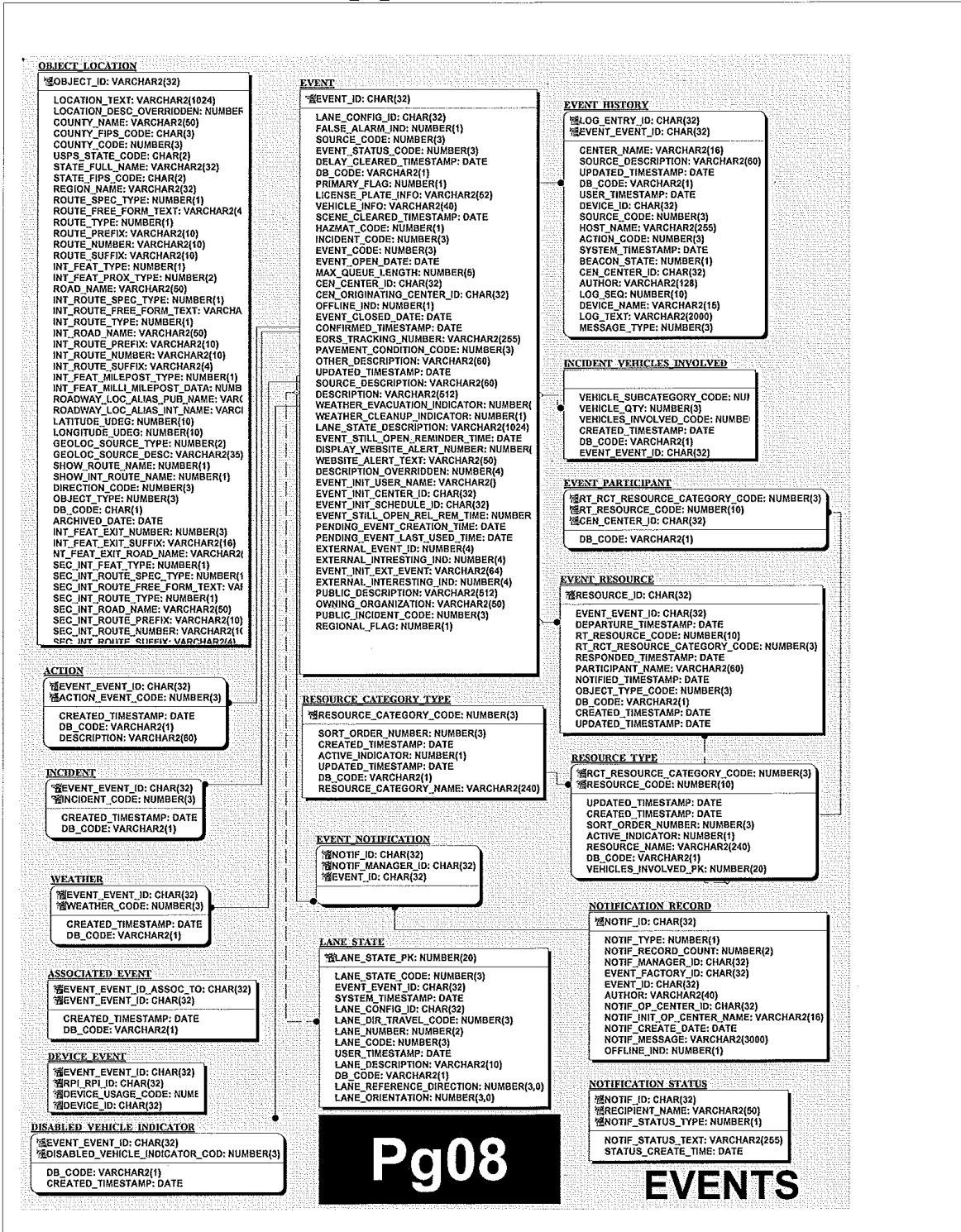
# CHART\_R7\_ERWIN73 -- CHART R7 / CHART MAIN





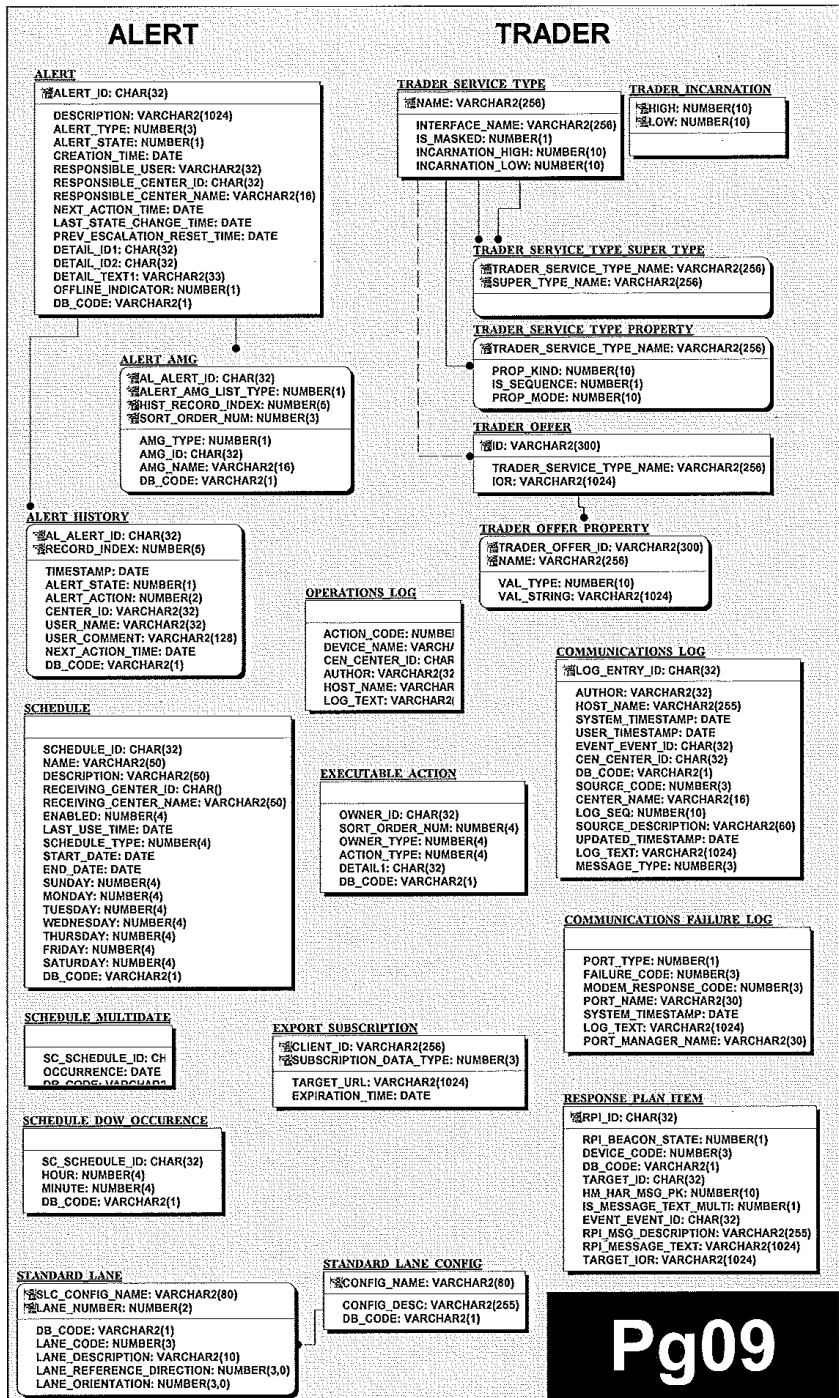






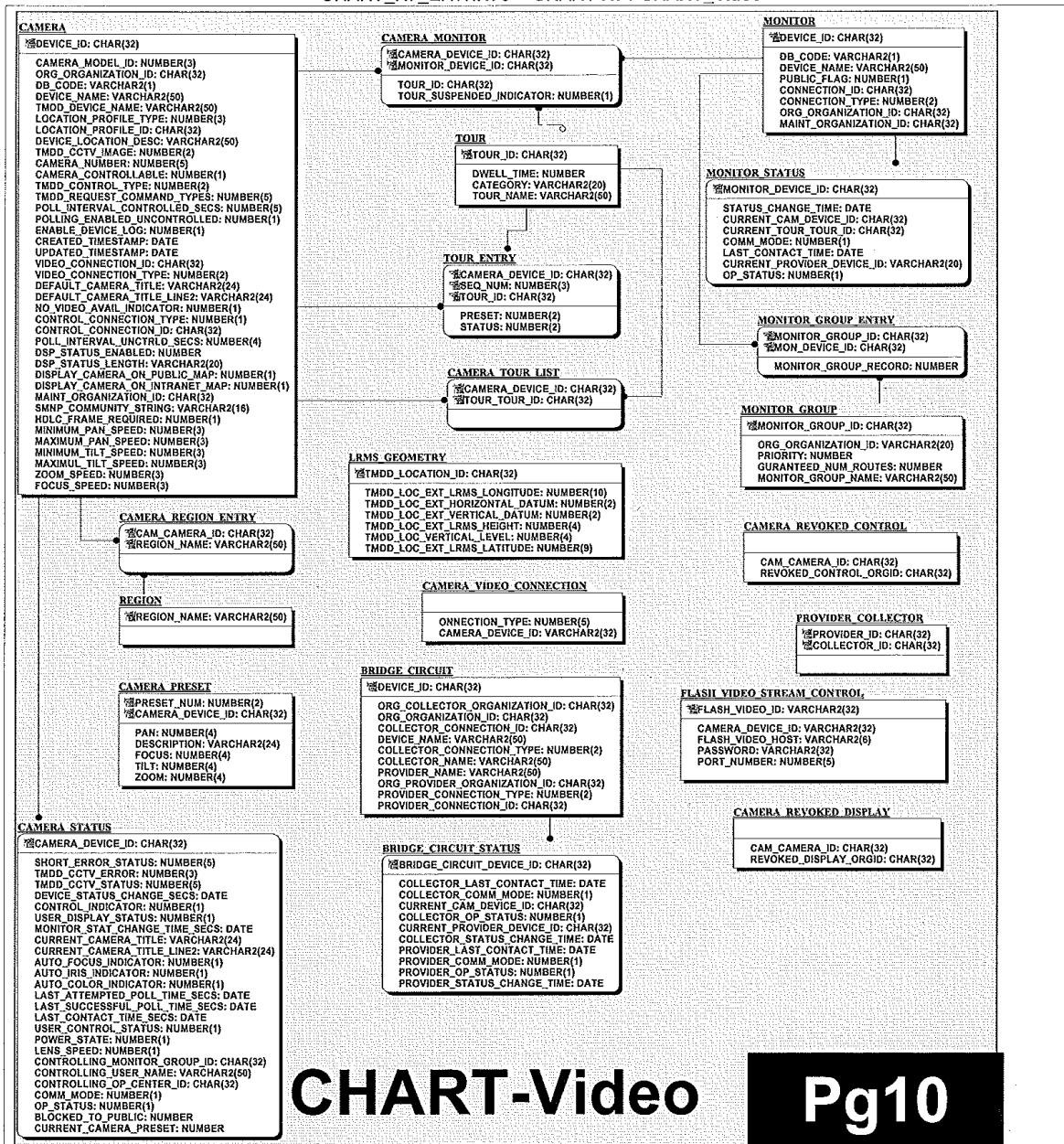
Pg08

EVENTS



Pg09

CHART\_R7\_ERWIN73 -- CHART R7 / CHART\_Video



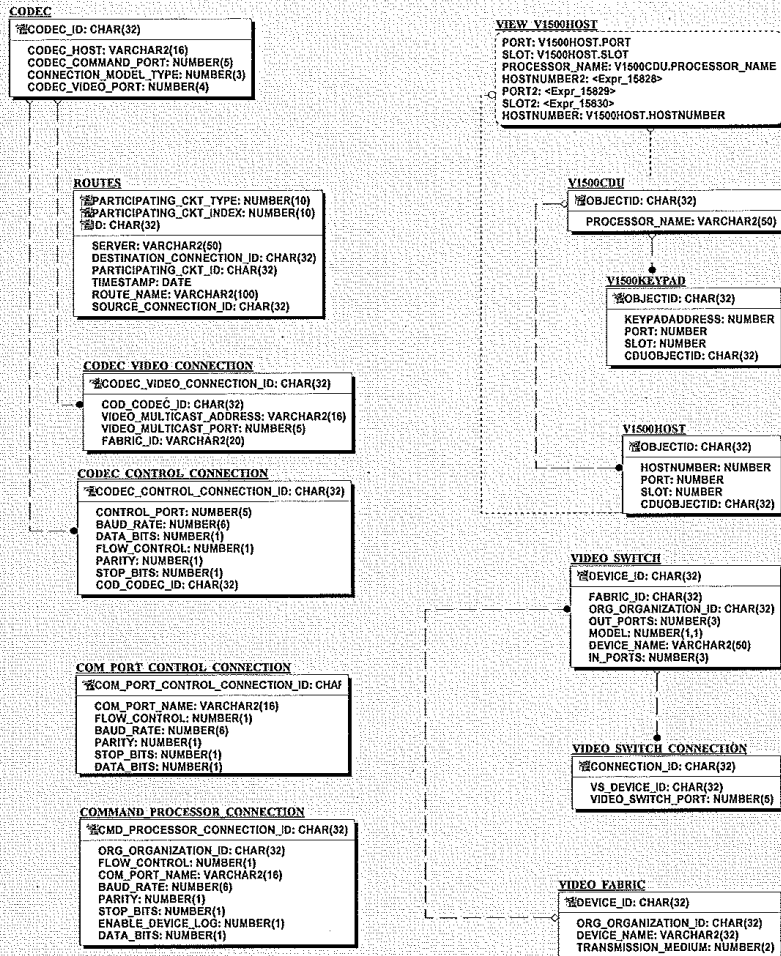
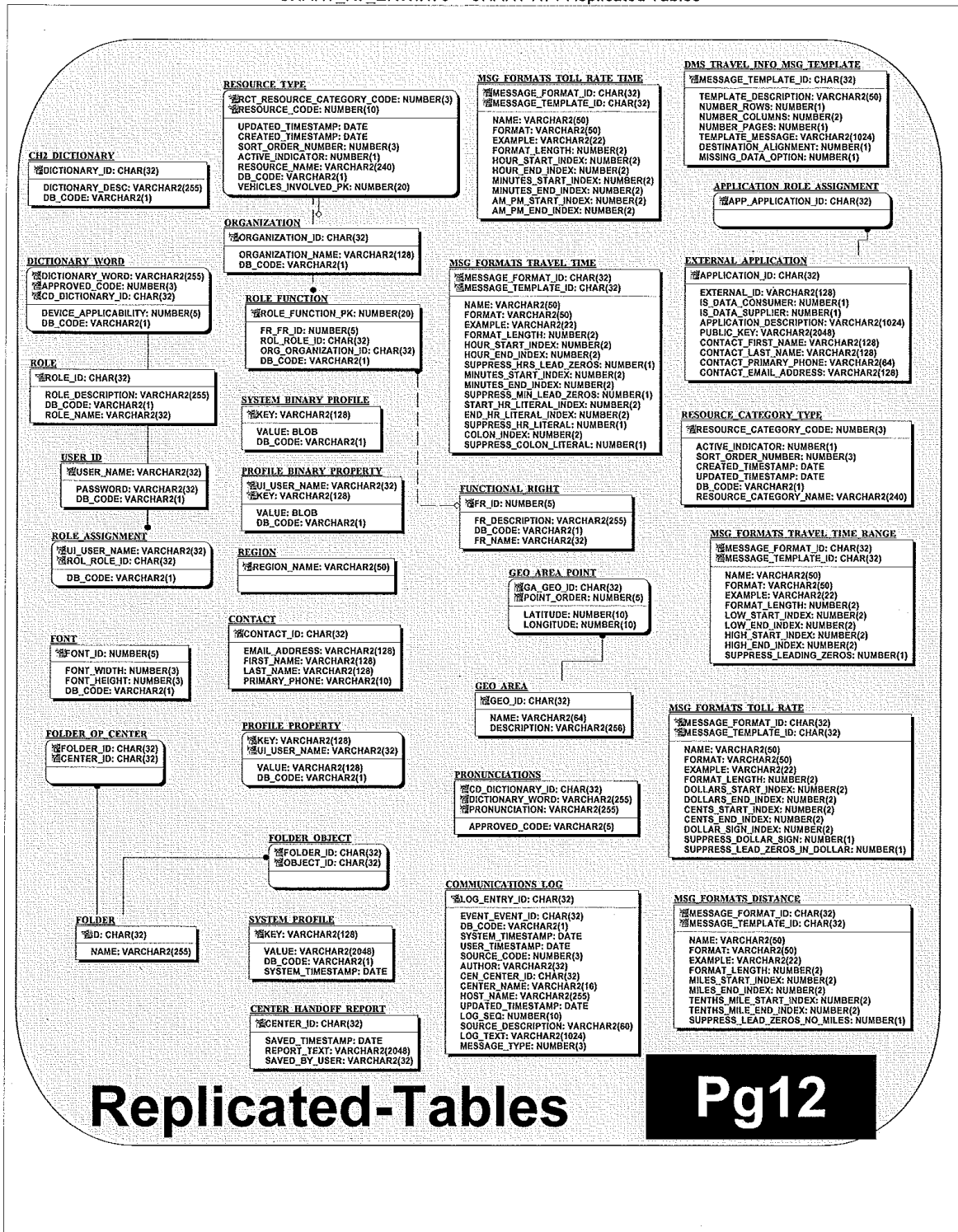


CHART-Video

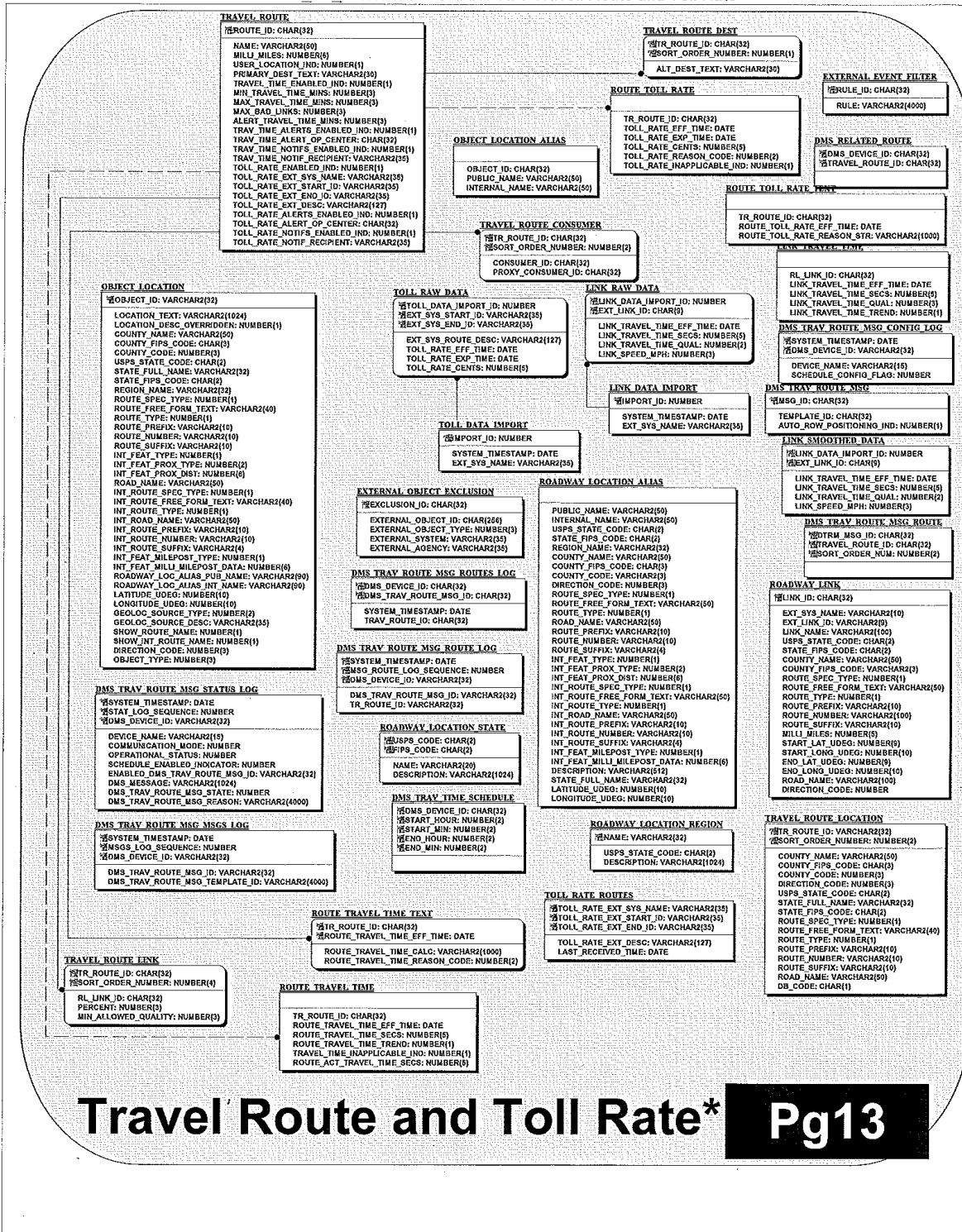
Pg11



Replicated-Tables

Pg12

CHART\_R7\_ERWIN73 -- CHART R7 / Travell Route and Toll Rate



Travel Route and Toll Rate\* Pg13

1, 1 / 1, 2 -- 3:54:58 PM, 2/17/2011

CHART R7 Detailed Design

#### 2.4.1.1.2.2 Function to Entity Matrix Report

The Create, Retrieve, Update, Delete (CRUD) matrix cross-references business functions to entities and shows the use of the entities by those functions. This report will be generated as part of the CHART O&M Guide.

#### 2.4.1.1.2.3 Table Definition Report –

In existing tables shown below:

- Deleted columns/constraints marked with a minus sign (“-”)
- Modified columns/constraints marked with an asterisk (“\*”)
- New columns/constraints marked with a plus sign (“+”)

#### 2.4.1.1.2.3.1 Tables Modified for the Integrated Map – Detector Bearing feature

##### 2.4.1.1.2.3.1.1 CHART

###### TSS Table

```
(+) DISPLAY_BEARING NUMBER(10) DEFAULT -1 NOT NULL
(+) CONSTRAINT display_bearing_ck CHECK (DISPLAY_BEARING BETWEEN -1 and
359)
```

###### TSS Zone Group Table

```
(+) DISPLAY_TYPE NUMBER(1) DEFAULT 0 NOT NULL,
(+) DISPLAY_ORDER NUMBER(10) DEFAULT 1 NOT NULL,
(+) CONSTRAINT display_type_ck CHECK (DISPLAY_TYPE BETWEEN 0 and 2).
```

#### 2.4.1.1.2.3.1.2 Mapping

##### **tss\_devices (table)**

“TSSSiteID” column is dropped.

##### **tss\_zones (table)**

Integer column “DisplayOrder” is added to indicate the zone group relative to the other zone groups of the same detector that are oriented the same way. Lower values indicate that this zone group is closer to the tss lat/lon position while larger values indicate that this zone group is further away.

“DirectionText” is dropped. Use “description” instead.

Bit field column “UpdateFlag” is added to indicate when the zone group record is newly added or when the bearing of the zone is updated.



## G\_TSSSITE (spatial table)

Instead of using previously defined “TSSSiteID” as the identifier, a unique 32 characters device id which is used by CHART will be used as the identifier.

### 2.4.1.1.2.3.2 Tables Modified for the NTCIP Camera feature

The NTCIP Camera feature requires changes to the CAMERA table. R7 will add ten new columns to the Camera table. Theses are marked with plus signs (“+”) in the following table:

Name	Null?	Type
-----	-----	-----
DEVICE_ID	NOT NULL	CHAR (32)
CAMERA_MODEL_ID	NOT NULL	NUMBER (3)
ORG_ORGANIZATION_ID	NOT NULL	CHAR (32)
DEVICE_NAME	NOT NULL	VARCHAR2 (50)
LOCATION_PROFILE_TYPE		NUMBER (3)
LOCATION_PROFILE_ID		CHAR (32)
TMDD_CCTV_IMAGE		NUMBER (2)
CAMERA_NUMBER		NUMBER (5)
CAMERA_CONTROLLABLE	NOT NULL	NUMBER (1)
TMDD_CONTROL_TYPE		NUMBER (2)
TMDD_REQUEST_COMMAND_TYPES	NOT NULL	NUMBER (5)
ENABLE_DEVICE_LOG	NOT NULL	NUMBER (1)
OLD_VIDEO_CONNECTION_ID		VARCHAR2 (32)
OLD_VIDEO_CONNECTION_TYPE		NUMBER (2)
NO_VIDEO_AVAIL_INDICATOR	NOT NULL	NUMBER (1)
DEVICE_LOCATION_DESC		VARCHAR2 (50)
TMDD_DEVICE_NAME		VARCHAR2 (50)
POLL_INTERVAL_CONTROLLED_SECS		NUMBER (5)
POLLING_ENABLED_UNCONTROLLED		NUMBER (1)
DEFAULT_CAMERA_TITLE		VARCHAR2 (24)
DEFAULT_CAMERA_TITLE_LINE2		VARCHAR2 (24)
CONTROL_CONNECTION_TYPE		NUMBER (1)
CONTROL_CONNECTION_ID		CHAR (32)
POLL_INTERVAL_UNCTRLD_SECS		NUMBER (4)
DB_CODE		VARCHAR2 (1)
CREATED_TIMESTAMP		DATE
UPDATED_TIMESTAMP		DATE
DSP_STATUS_ENABLED		NUMBER (1)
DSP_STATUS_LENGTH		NUMBER (5)
DISPLAY_CAMERA_ON_PUBLIC_MAP		NUMBER (1)
DISPLAY_CAMERA_ON_INTRANET_MAP		NUMBER (1)
MAINT_ORGANIZATION_ID		CHAR (32)
<b>+SMNP_COMMUNITY_STRING</b>		<b>VARCHAR2 (30)</b>
<b>+HDLC_FRAME_REQUIRED</b>		<b>NUMBER (1)</b>
<b>+MINIMUM_PAN_SPEED</b>		<b>NUMBER (3)</b>
<b>+MAXIMUM_PAN_SPEED</b>		<b>NUMBER (3)</b>
<b>+MINIMUM_TILT_SPEED</b>		<b>NUMBER (3)</b>

<b>+MAXIMUM_TILT_SPEED</b>	<b>NUMBER (3)</b>
<b>+ZOOM_SPEED</b>	<b>NUMBER (3)</b>
<b>+FOCUS_SPEED</b>	<b>NUMBER (3)</b>
<b>+MIN_ZOOM_POSITION</b>	<b>NUMBER (5)</b>
<b>+MAX_ZOOM_POSITION</b>	<b>NUMBER (5)</b>

#### 2.4.1.1.2.3.3 Tables Modified for the SCAN Weather Integration Feature

The weather information will be stored in a single new field in the EVENT table, in JSON format to represent the data for the selected weather station and sensor data.

Name	Null?	Type
EVENT_ID	NOT NULL	CHAR (32)
LANE_CONFIG_ID		CHAR (32)
DB_CODE		VARCHAR (1)
EVENT_CODE	NOT NULL	NUMBER (3)
EORS_TRACKING_NUMBER		VARCHAR2 (255)
CEN_CENTER_ID		CHAR (32)
CEN_ORIGINATING_CENTER_ID		CHAR (32)
PRIMARY_FLAG		NUMBER (1)
LICENSE_PLATE_INFO		VARCHAR2 (52)
VEHICLE_INFO		VARCHAR2 (40)
OFFLINE_IND		NUMBER (1)
MAX_QUEUE_LENGTH		NUMBER (5)
EVENT_STATUS_CODE		NUMBER (3)
SCENE_CLEARED_TIMESTAMP		DATE
DELAY_CLEARED_TIMESTAMP		DATE
CONFIRMED_TIMESTAMP		DATE
FALSE_ALARM_IND		NUMBER (1)
EVENT_CLOSED_DATE		DATE
EVENT_OPEN_DATE		DATE (7)
SOURCE_CODE		NUMBER (3)
HAZMAT_CODE		NUMBER (1)
INCIDENT_CODE		NUMBER (3)
WEATHER_CLEANUP_INDICATOR		NUMBER (1)
WEATHER_EVACUATION_INDICATOR		NUMBER (1)
PAVEMENT_CONDITION_CODE		NUMBER (3)
UPDATED_TIMESTAMP		DATE (7)
OTHER_DESCRIPTION		VARCHAR2 (60)
DESCRIPTION		VARCHAR2 (512)
SOURCE_DESCRIPTION		VARCHAR2 (60)
LANE_STATE_DESCRIPTION		VARCHAR2 (1024)
EVENT_STILL_OPEN_REMINDER_TIME		DATE
DISPLAY_WEBSITE_ALERT		NUMBER (1)
WEBSITE_ALERT_TEXT		VARCHAR2 (3000)
DESCRIPTION_OVERRIDDEN		NUMBER (1)
AUX_DESCRIPTION		VARCHAR2 (512)
EVENT_INIT_USER_NAME		VARCHAR2 (40)
EVENT_INIT_CENTER_ID		CHAR (32)
EVENT_INIT_SCHEDULE_ID		CHAR (32)
EVENT_INIT_EXT_SYSTEM		VARCHAR2 (35)
EVENT_INIT_EXT_AGENCY		VARCHAR2 (35)
EVENT_INIT_EXT_EVENT		VARCHAR2 (35)

EVENT_STILL_OPEN_REL_REM_TIME	NUMBER (8)
PENDING_EVENT_CREATION_TIME	DATE
PENDING_EVENT_LAST_USED_TIME	DATE
EXTERNAL_EVENT_IND	NUMBER (1)
EXTERNAL_INTERESTING_IND	NUMBER (1)
PUBLIC_DESCRIPTION	VARCHAR2 (512)
OWNING_ORGANIZATION	CHAR (32)
PUBLIC_INCIDENT_CODE	NUMBER (3)
REGIONAL_FLAG	NUMBER (1)
<b>(+)WEATHER_INFO_JSON</b>	<b>VARCHAR2 (1024)</b>

#### **2.4.1.1.2.4 PL/SQL Module Definition and Database Trigger Reports**

There are no new PL/SQL modules for CHART R7.

#### **2.4.1.1.2.5 Database Size Estimate - provides size estimate of current design**

There are no changes for any significance to the database size for R7.

#### **2.4.1.1.2.6 Data Distribution**

There are no changes to data distribution for R7.

#### **2.4.1.1.2.7 Database Replication**

There are no changes to database replication for R7.

#### **2.4.1.1.2.8 Archival Migration**

There are no changes to archival migration for R7.

#### **2.4.1.1.2.9 Database Failover Strategy**

There are no changes to the database failover strategy for R7.

#### **2.4.1.1.2.10 Reports**

No reports will be added or updated for R7. Since R5, the CHART reporting function has been transferred to University of Maryland.

### **2.4.1.2 CHART Flat Files**

The following describes the use of flat files in CHART.

#### **2.4.1.2.1 Service Registration Files**

There are no new Java services and therefore no new service registration files for CHART R7.

#### **2.4.1.2.2 Service Property Files**

Except as noted, there are no new service property files for CHART R7.

#### **2.4.1.2.3 GUI Property Files**

There are only minor updates to the GUI properties file in its WEB-INF directory for CHART R7.

#### **2.4.1.2.4 Arbitration Queue Storage Files**

There are no changes to Arbitration Queue Storage Files for R7.

#### **2.4.1.2.5 Device Logs**

There are no changes to Device Log Files for R7.

#### **2.4.1.2.6 Traffic Sensor Raw Data Logs**

There are no changes to Traffic Sensor Raw Data Log Files for R7.

#### **2.4.1.2.7 Service Process Logs**

All CHART services write to a process log, used to provide a historical record of activity undertaken by the services. These logs are occasionally referenced by software engineering personnel to diagnose a problem or reconstruct a sequence of events leading to a particular anomalous situation. These logs are automatically deleted by the system after a set period of time defined by the service's properties file, so they do not accumulate infinitely. These files are stored in the individual service directories and are named by the service name and date, plus a ".txt" extension. These logs are typically read only by software engineering personnel. Except where noted, there are no changes for service process logs for R7 features.

#### **2.4.1.2.8 Service Error Logs**

All CHART services write to an error log, used to provide detail on certain errors encountered by the services. Most messages, including most errors, are captured by the CHART software and written to the process logs, but certain messages (typically produced by the Java Virtual Machine itself, by COTS, or DLLs) cannot be captured by CHART Software and instead are captured in these "catch-all" logs. Errors stored in these logs are typically problems resulting from a bad installation; once the system is up and running, errors rarely appear in these error logs. Debugging information from the JacORB COTS, which is not usually indicative of errors, can routinely be found in these error logs, as well. These log files can be reviewed by software engineering personnel to diagnose an installation problem or other type of problem. These logs are automatically deleted by the system after a set period of time defined by the service's properties file, so they do not accumulate infinitely. These files are stored in the individual service directories and are named by the service name and date, plus an ".err" extension. These logs are typically read only by software engineering personnel. Except where noted, there are no changes for service error logs for R7 features.

#### **2.4.1.2.9 GUI Process Logs**

Like the CHART background services, the CHART GUI service also writes to a process log file, used to provide a historical record of activity undertaken by the process. These GUI process logs are occasionally referenced by software engineering personnel to diagnose a problem or reconstruct a sequence of events leading to a particular anomalous situation. These logs are

automatically deleted by the system after a set period of time defined by the GUI service's properties file, so they do not accumulate infinitely. These files are stored in the `chartlite/LogFiles/` directory under the `WebApps/` directory in the Apache Tomcat installation area. They are named by the service name ("chartlite") and date, plus a ".txt" extension. These logs are typically read only by software engineering personnel. Additional log files written by the Apache Tomcat system itself are stored in the `log/` directory in the Apache Tomcat installation area.

- R7 GUI changes do not change the way the GUI process logs operate.

#### **2.4.1.2.10 FMS Port Configuration Files**

The CHART Communications Services read a Port Configuration file, typically named `PortConfig.xml`, upon startup, which indicates which ports are to be used by the service and how they are to be initialized. A Port Configuration Utility is provided which allows for addition, removal of ports and editing of initialization parameters. As indicated by the extension, these files are in XML format. This means these files are hand-editable, although the Port Configuration Utility allows for safer, more controlled editing. The Port Configuration files are typically modified only by software engineers or telecommunications engineers.

- There are no changes to this section for the any of the R7 features.

#### **2.4.1.2.11 Watchdog Configuration Files**

The watchdog configuration files are updated to provide monitoring and restarting of the CHART Lane Configuration Editor Web Service and User Management Web Service.

### **2.4.2 Database Design**

Changes made to the CHART database design for Release 7 features are described below.

#### **2.4.2.1 Integrated Map – Detector Bearing**

##### **2.4.2.1.1 CHART**

- Adds `DISPLAY_BEARING` column to the TSS table. This column is constrained to integer values between -1 and 359. A value of -1 indicates that the TSS has no defined bearing. A value between 0 and 359 indicates the bearing rotation for the TSS where a value of 0 means due east and values grow counter clockwise such that a value of 90 is due north and a value of 359 is 1 degree South of due east.
- Adds `DISPLAY_TYPE` column to the `TSS_ZONE_GROUP` table. This column is constrained to the values 0 (do not display), 1 (display using arrow that points to TSS display bearing) and 2 (display using arrow that points 180 degrees opposed to TSS bearing).
- Adds `DISPLAY_ORDER` column to the `TSS_ZONE_GROUP` table. This column provides an ordinal that allows the zone group to be ordered relative to other zone groups that have the same display type. Lower values are displayed closer to the TSS lat/lon position and higher values are displayed further away from the TSS coordinate.

-A new script will be implemented to migrate zone group bearings from the CHARTWeb database into the CHART database.

#### **2.4.2.1.2 Intranet Mapping**

##### **tss\_devices (table)**

“TSSSiteID” column is dropped.

##### **tss\_zones (table)**

Integer column “DisplayOrder” is added to indicate the zone group relative to the other zone groups of the same detector that are oriented the same way. Lower values indicate that this zone group is closer to the tss lat/lon position while larger values indicate that this zone group is further away.

“DirectionText” is dropped. Use “description” instead.

Bit field column “UpdateFlag” is added to indicate when the zone group record is newly added or when the bearing of the zone is updated.

##### **G\_TSSSITE (spatial table)**

Instead of using previously defined “TSSSiteID” as the identifier, a unique 32 characters device id which is used by CHART will be used as the identifier.

##### **GV\_TSS (view)**

GV\_TSS is updated to reflect changes made in G\_TSSSITE, tss\_devices and tss\_zones tables.

##### **vw\_TSS (view)**

vw\_TSS is updated to reflect changes made in G\_TSSSITE, tss\_devices and tss\_zones tables.

##### **tss\_add\_zone.sql (store procedure)**

Additional process is to add the tss\_add\_zone store procedure to determine when a tss zone is newly added or when the bearing of the zone is updated.

##### **tss\_remap\_site\_id\_wl.sql (store procedure)**

tss\_remap\_site\_id\_wl.sql is updated to reflect changes made in tss\_devices table.

##### **tss\_remap\_site\_zones\_wl.sql (store procedure)**

tss\_remap\_site\_zones\_wl.sql is updated to reflect changes made in tss\_zones table.

#### **Mobile\_Count\_Home.sql (store procedure)**

Mobile\_Count\_Home.sql is updated to reflect changes made in G\_TSSSITE table.

#### **Mobile\_Tss\_Data.sql (store procedure)**

Mobile\_Tss\_Data.sql is updated to reflect changes made in G\_TSSSITE table.

#### **fn\_TSS\_Tooltip.sql (function)**

fn\_TSS\_Tooltip.sql is updated to reflect changes made in vw\_TSS view.

#### **fn\_TSSRange\_Tooltip.sql (function)**

fn\_TSSRange\_Tooltip.sql is updated to reflect changes made in vw\_TSS view.

### **2.4.2.2 NTCIP Camera Support**

The CAMERA table in the CHART R7 database will have 10 new columns as shown above.

### **2.4.2.3 SCAN Weather Integration**

The SCAN Weather Integration feature will utilize an existing table in the CHARTWeb MSSQL Database and will require the addition of several new views and tables as described below: The CHART Oracle DB will have minor modifications as described above.

#### **2.4.2.3.1 CHART DB**

The weather info will be stored in a single new field in the EVENT table, in JSON format to represent the data for the selected weather station and sensor data. That change is described above. Since this data is only being displayed in the CHART GUI and is not being used otherwise, a more structured format (i.e., multiple fields and an additional table for sensor data) was not required at this time and this mechanism allows for flexibility. If the fields change in the future, only the internal format and the GUI would need to change. A version number will be encoded within the JSON and the format will be documented (perhaps in the TrafficEventManager IDL comments) to avoid backward compatibility issues if the internal format needs to change.

#### **2.4.2.3.2 CHARTWeb DB**

##### **CHARTWeb DB G\_RWIS Table:**

This is an existing table currently in use by the map application. For the SCAN Weather integration it is used to specify which weather stations will be available in CHART and to provide the geo-location of the Weather Station itself. No changes are needed to this table for the SCAN Weather Integration feature.

[OBJECTID] [int] NOT NULL

[NAME] [varchar](255)  
 [SYS\_NUMBER] [smallint] NOT NULL  
 [RPU\_NUMBER] [smallint] NOT NULL  
 [HASCAMERA] [int]  
 [HASSCANCAST] [int]  
 [X] [float]  
 [Y] [float]  
 [HASAERIALPHOTO] [int] NOT NULL  
 [AERIALPHOTOX] [float]  
 [AERIALPHOTOY] [float]  
 [AERIALPHOTOWIDTH] [int]  
 [AERIALPHOTOHEIGHT] [int]  
 [UPDATETIME] [datetime]  
 [LATITUDE] [float]  
 [LONGITUDE] [float]  
 [SHAPE] [int]  
 [county\_ID\_1] [int]  
 [county\_ID\_2] [int]

#### **CHARTWeb DB RWIS\_Sensor\_Route\_lkp Table:**

This table is new for R7 and is used to support the SCAN Weather Integration. The table is populated with location information for each Weather Station's surface sensors. This table contains location information not available in SCAN including route prefix, route number, route suffix and direction. This additional location information is used when determining the best surface sensor to use when determining surface condition reported at the Weather Station level.

[sys\_number] [smallint] NOT NULL,  
 [rpu\_number] [smallint] NOT NULL  
 [sensor\_number] [smallint] NOT NULL,  
 [route\_prefix] [varchar](2)  
 [route\_num] [varchar](7)  
 [route\_suffix] [varchar](2)  
 [route\_direction] [varchar](20)  
 [sensor\_name] [varchar](35)

#### **CHARTWeb DB RWIS\_surface\_conditions\_lkp table:**

This table is new for R7 to support the SCAN Weather Integration. It is used to map SCAN road condition values to chart RoadCondition values (WeatherService.xsd).

[chart\_value] [varchar](35)  
 [scan\_value] [varchar](35)



### **CHARTWeb DB RWIS\_AtmosphericReading View:**

This view is new for R7 to support the SCAN Weather Integration. It is used to provide current atmospheric data from the SCAN DB. It joins the G\_RWIS table in the CHARTWeb DB with the AtmosphericReading view in the SCAN DB. This view will be queried every N minutes (configurable) to retrieve current atmospheric data. Join defined using the following FROM clause: FROM G\_RWIS a INNER JOIN [SOC-SCANDB-SVR].[external].dbo.AtmosphericReading b ON b.sysid = a.Sys\_Number AND b.rpuid = a.Rpu\_Number.

```
a.Sys_Number AS sys_number[smallint] NOT NULL
a.Rpu_Number AS rpu_number[smallint] NOT NULL
a.name[varchar](255)
b.AirTemperature AS air[smallint]
b.DewPoint AS dew[smallint]
b.RelativeHumidity AS rh[smallint]
b.WindSpeed AS spd_avg[smallint]
b.WindDirection AS dir_avg[varchar](25)
b.WindGust AS spd_gst,[smallint]
b.WindGustDirection AS dir_gst[varchar](25)
b.Visibility AS vis [smallint]
b.PrecipitationType AS precip [varchar](25)
b.PrecipitationIntensity AS intens[varchar](25)
b.PrecipAccum AS accum[smallint]
b.ReadingDateTime AS datetime[date]
```

### **CHARTWeb DB SurfaceReading View:**

This view is new for R7 to support the SCAN Weather Integration. It is used to provide current surface data from the SCAN DB. It joins the G\_RWIS table in the CHARTWeb DB with the SurfaceReading view in the SCAN DB. This view will be queried every N minutes (configurable) to retrieve current surface data. Join defined using the following FROM clause: FROM G\_RWIS a INNER JOIN [SOC-SCANDB-SVR].[external].dbo.SurfaceReading b ON b.sysid = a.Sys\_Number AND b.rpuid = a.Rpu\_Number

```
a.Sys_Number AS sys_number[smallint] NOT NULL
a.Rpu_Number AS rpu_number[smallint] NOT NULL
b.SurfaceSensorid AS sensor_number[smallint] NOT NULL
b.SensorName[varchar](255)
b.ReadingDateTime AS datetime,[date]
b.surfacecondition AS status[varchar](255)
b.surfacetemperature AS sfcTemp[smallint]
```

#### **2.4.2.4 Archiving - Changes**

The CHART Archive database stores data from the CHART operational system as part of a permanent archive. The CHART Archive database design is a copy of the CHART operational system for those tables containing system, alert, traveler information messages and their underlying data, and event log information. In addition, the CHART Archive database stores detector data. In CHART R7 archive database changes need to be made on the EVENT table. There are TSS\_DEVICE and CAMERA Related changes in the DEVICE table. GETOPERATIONALDATA needs modifications and testing as well.

## 3 Key Design Concepts

---

### 3.1 Integrated Map – Detector Bearing

#### Configuration

Changes have been made to the TSS details pages to allow an administrator to configure the map display options for a TSS. These changes can be made regardless of the TSS mode (online, offline, maintenance mode) and can be made to external detectors that have been imported into CHART as well as native CHART detectors. The Map display options pages allow a user to specify the direction that the arrows for a TSS should be oriented when displaying on the map. Changes have also been made to allow the user to indicate if each zone group should be displayed using an arrow that points toward the TSS bearing, using an arrow that points 180 degrees opposite the TSS bearing, or should not be displayed at all. If a TSS has multiple zone groups that are configured to display in the same direction the CHART administrator may configure their relative display order so that zone groups that represent outer lanes can be displayed further away from the TSS lat/lon position and zone groups that represent inner lanes can be displayed closer.

#### Map Display

When displaying a TSS on any of the GUI integrated maps, the system will always display an icon on the map for any TSS that has a defined location. If a bearing has been defined and at least one zone group is configured for map display and the TSS is online the system will render the TSS on the map using an arrow for each configured zone group. The color of each arrow will represent current speed for the zone group it represents. If any of those conditions are not met, the TSS will be displayed using the same icon that is used in the GUI list pages for the TSS. When a TSS is added to the system, either via import from an external system or via an administrator action, it will be added with no defined bearing. This implies that the system will display the TSS on maps using the list icon until an administrator manually sets the bearing. The system will inspect the TSS route direction from its location settings and also inspect the direction of each configured zone group and set the display type to primary if the direction matches, opposite if the direction is opposite, and do not display on maps if the direction of the zone group is neither the same nor opposite the direction of the TSS route. The system will not attempt to default the display order if there are multiple zone groups in a single direction. Using this algorithm it is anticipated that an administrator will usually need only set the TSS location, zone group directions, then bearing in order to get the TSS to properly display on the map. If external systems do not provide the route direction location information for a TSS the administrator will have to set the TSS bearing, then set the display type (primary, opposite, none) for each zone group in order to get the TSS to display properly using arrows.

#### Key Design Decisions

Because there are many TSS objects that will need to be rendered on maps, performance related decisions were key in the design of this feature.

- In order to avoid overloading the mapping server the TSS objects are rendered client side.

- Because there will be over 400 TSS objects to render when viewing the entire state and surrounding areas, and because browser recommendations are that no more than 50-100 markers should be rendered client side (both to avoid clutter and to avoid CPU overload on the browser machine), the system will only allow the TSS layers to be visible at configured map scales (zoom levels).
- Because the client browser has to request the data used to render the TSS objects using AJAX requests that return data in JSON format (ASCII Text), which the browser must parse into JavaScript objects, it is important to limit the size of the JSON documents being returned to the browser each time the user needs their map display updated. To that end, the system will only request JSON for objects within the visible display area of the user's current map. This implies that when the user pans the map to include a previously unviewed area, or zooms the map out (so that more area becomes visible around the previously viewed area) there will be a delay while the JSON data for the objects in the newly visible are retrieved from the web server, parsed, and rendered.

## **3.2 NTCIP Camera Support**

The NTCIP Camera feature builds upon the support that already exists in CHART for Vicor and Cohu cameras. A new protocol handler will be created to handle the translation from application level actions such as Pan and Zoom to NTCIP compliant commands to be delivered to the camera controller. Changes will be made to allow some re-use of utility code that already exists for NTCIP communications to DMS devices. One area where control of NTCIP cameras differs from the currently supported camera models is that the NTCIP protocol requires a speed parameter on every movement command. Rather than supporting variable speed movement via GUI controls, the approach is to allow default speed values to be set for each camera for each of 4 movements: pan, tilt, zoom, and focus. Each camera will have 1 speed setting for Zoom and 1 speed setting for Focus. A single speed value for pan and tilt is not sufficient, however, because when zoomed in you need the camera to move slower than when zoomed out. To accomplish this feature there will be two speed settings for each camera for Pan (min and max), and two settings for Tilt (min and max). When a pan or tilt operation is performed, CHART will determine the speed to use based on the min and max speed settings and the current zoom level of the camera. Two additional settings, min zoom value and max zoom value will also need to be configured for each camera to allow for this zoom based variable speed behavior.

An NTCIP Camera Compliance Tester is also included as part of R7. The purpose of this application is to allow camera vendors to verify that their NTCIP camera will work with the CHART system. The design of this tester is based on the design of the NTCIP DMS Compliance tester. It differs in the user commands it supports and the protocol handler it uses, however the design of the basic framework of the tester, such as the base application and communications features are similar.

## **3.3 SCAN Weather Integration**

A new CHART Web Service will be created in order to provide internal CHART applications with weather related data. This web service will currently retrieve Weather Station data from the SCAN system and will provide it to internal CHART application in XML form. The XML

schema defining this interface was designed to be generic and not tied to any one provider of Weather Information (I.E. not dependent on the SCAN system currently used).

The CHART GUI and Traffic Event Service will be modified to use this new Web Service to allow pre-population of Traffic Event Road Conditions where applicable and provide display of other weather details for a Traffic Event. This is discussed further in the Human Machine Interface section.

For Release 7, the integration of the SCAN Weather system into CHART is primarily for operator convenience however the design anticipates a future expanded role by defining an extensible message set and isolating all SCAN-specific fields and parameters into one area of the Weather web service.

- The design includes a modular approach to weather data sources. The addition of a new data source requires only the addition of an interface to the new source in the web service. This could be a new DB connection, a flat file, access to another web service or similar format. The rest of the architecture remains unchanged.
- The design includes a modular approach to weather data clients. If the new client requires access to summary weather information (air temperature, wind speed and direction, surface temperature, surface conditions, or precipitation) then no changes to the weather web service is required. More detailed weather information such as relative humidity, dew point, trends, and historical reports will require changes to the messaging though no changes to the architecture.

### **3.4 Shift Handoff Report**

A new Shift Handoff Report is included as part of Release 7. The Shift Handoff Report is a means of sharing important operations information with operators. The report is provided using a third part product, WordPress, and as such does not contain any design elements within this document. WordPress provides the ability to include dynamic information to operators in the form of blog posts and static reference information in the form of web pages. A built-in menu system allows operators to navigate through the site to view both types of information. Existing Shift Handoff links in the CHART system are updated to point to this new Shift Handoff Report.

### **3.5 Error Processing**

In general, CHART traps conditions at both the GUI and at the server. User errors that are trapped by the GUI are reported immediately back to the user. The GUI will also report communications problems with the server back to the user. The server may also trap user errors and those messages will be written to a server log file and returned back to the GUI for display to the user. Additionally, server errors due to network errors or internal server problems will be written to log files and returned back to the GUI.

## 3.6 Packaging

### 3.6.1 CHART

This software design is broken into packages of related classes. The table below shows each package that is new or changed to support the Release 7 features.

Package Name	Package Description
<b>CHART2.CameraControlModule</b>	This package is changed for R7 to support the NTCIP Camera model.
<b>CHART2.DeviceUtility</b>	This package is changed for R7 to refactor some of the communication related utilities allow their use with the NTCIP Camera Protocol Handler.
<b>CHART2.TrafficEventManager</b>	This CORBA package will be modified to add a weather conditions field to both IncidentData and WeatherServiceEventData.
<b>CHART2.TrafficEventModule</b>	This Traffic Event Service package will be changed to support querying the Weather Service.
<b>CHART2.TSSManagementModule</b>	This TSS Service package has been modified to support the new setMapDisplayOptions() API.
<b>CHART2.TSSManagement</b>	This CORBA package has been modified to accommodate the TSS bearing, zone group display type, and zone group relatively display order. A new API has also been added to allow an administrator to set the map display options for a TSS without requiring them to set the entire configuration.
<b>CHART2.Utility</b>	This package will be changed to support a new utility for making time-limited synchronous queries. This will be used to limit the amount of time the opening of a traffic event will be delayed, if for some reason the call to the Weather Service does not complete quickly.
<b>chartlite.data</b>	This GUI package will be changed to add support for system profile properties to allow the administrator to set the weather station search radius and lookback time.
<b>chartlite.data.trafficevents</b>	This GUI package will be changed to add an accessor method to the WebIncident and WebWeatherServiceEvent to return the new IDL field. (NOTE – the formatting of the weather data will be done in the Javascript on the web page)
<b>chartlite.data.tss</b>	This GUI package has been changed to make the WebTSS class implement the MapFeature interface. This change will allow the WebTSS objects to be displayed on the integrated map.
<b>chartlite.data.video</b>	This is changed in R7 to support the NTCIP Camera model.
<b>chartlite.servlet.tss</b>	This GUI package has been modified to add methods that display the TSS Map Display Options page and process the changes from this page when it is submitted by a user.
<b>chartlite.servlet.map</b>	This GUI package has been modified to include processing that will add TSS layers to the home page map and nearby devices map.
<b>chartlite.servlet.usermgmt</b>	This GUI package will be changed to add support for system profile properties to allow the administrator to set the weather station search radius and lookback time.
<b>chartlite.servlet.video.source</b>	This package is changed in R7 to support the NTCIP Camera model.
<b>CHART.xsd.weatherservice</b>	This package is new for R7 and contains class generated from the WeatherService.xsd file.
<b>CHART.webservices.weathermodule</b>	This package is new for R7 and provides the implementation of the web service module making up the CHART2 WeatherService web service.

### 3.6.2 Mapping

The table below shows each package (Namespace in .Net) that is new or changed to support the Release 6 Integrated Map - Detector Bearing feature.

Namespace Name	Namespace Description
CHARTMap.Handlers	Existing namespace and extended for the class “TSSInventoryHandler” that comprise the TSS Data Synchronization in Mapping R6.

### 3.7 Assumptions and Constraints

1. Assumption that zone groups only need to point to the TSS bearing or directly opposite the TSS bearing. This assumption has been verified with Dale. The risk of needing other bearings is mitigated by the fact that we export the zone group display bearing as a bearing per zone group. This implies that the changes needed if we ever do need to support other bearings per zone group would be limited to the CHART TSS Service and GUI.
2. Assumption that we do not change the arrow direction for a zone group in an automated way even if the underlying lane(s) change directions. Zone group bearings can only be set toward or opposite the primary bearing of the TSS and must be modified by an administrator using the TSS map display options form.
3. Assumption: The Mapping TSS editing functionalities in the CHART Device Editor will be retired.
4. Assumption: Both the SCAN and CHART Web databases can support the periodic querying of a modest amount of data. The access rate for each is expected to be every few minutes for perhaps a few hundred records.
5. Constraint: The Intranet Map and SCAN web links on the Traffic Event Details page may show an error page if the browser is unable to reach those web services; perhaps because they are behind a firewall.
6. Assumption: The NTCIP Cameras deployed in the CHART system will support at least one of the two documented ways of determining the current zoom level of the camera. Without support for either of these methods, the system will not be able to provide variable speed pan and tilt based on zoom level. For cameras that do not support querying zoom level, the fallback position is to continue to control them with their manufacturer proprietary protocol. The NTCIP Camera Compliance Tester will include testing of the ability to query the zoom level, therefore this should not be an issue for cameras purchased in the future if the variable speed pan/tilt works properly via the tester.

#### Camera models tested successfully

Camera Model	Firmware
COHU 3955	IVIEW VER 1.15

	IVIEW VER 1.16 IVIEW VER 1.17 Rev B
COHU 3960	Rev B
Vicon SVFT	NTCIP

### **Camera Controllers Tested**

<u>Controller</u>	<u>Firmware</u>	<u>Zoom mib support</u>	<u>Position mib support</u>
COHU iControl 9300 Series	iDome NTCIP 1.0a 04/21/04		X
COHU iControl 9300 Series	1.5.7 Nov 23 2005	X	X



## **4 Use Cases – Integrated Map – Detector Bearing**

---

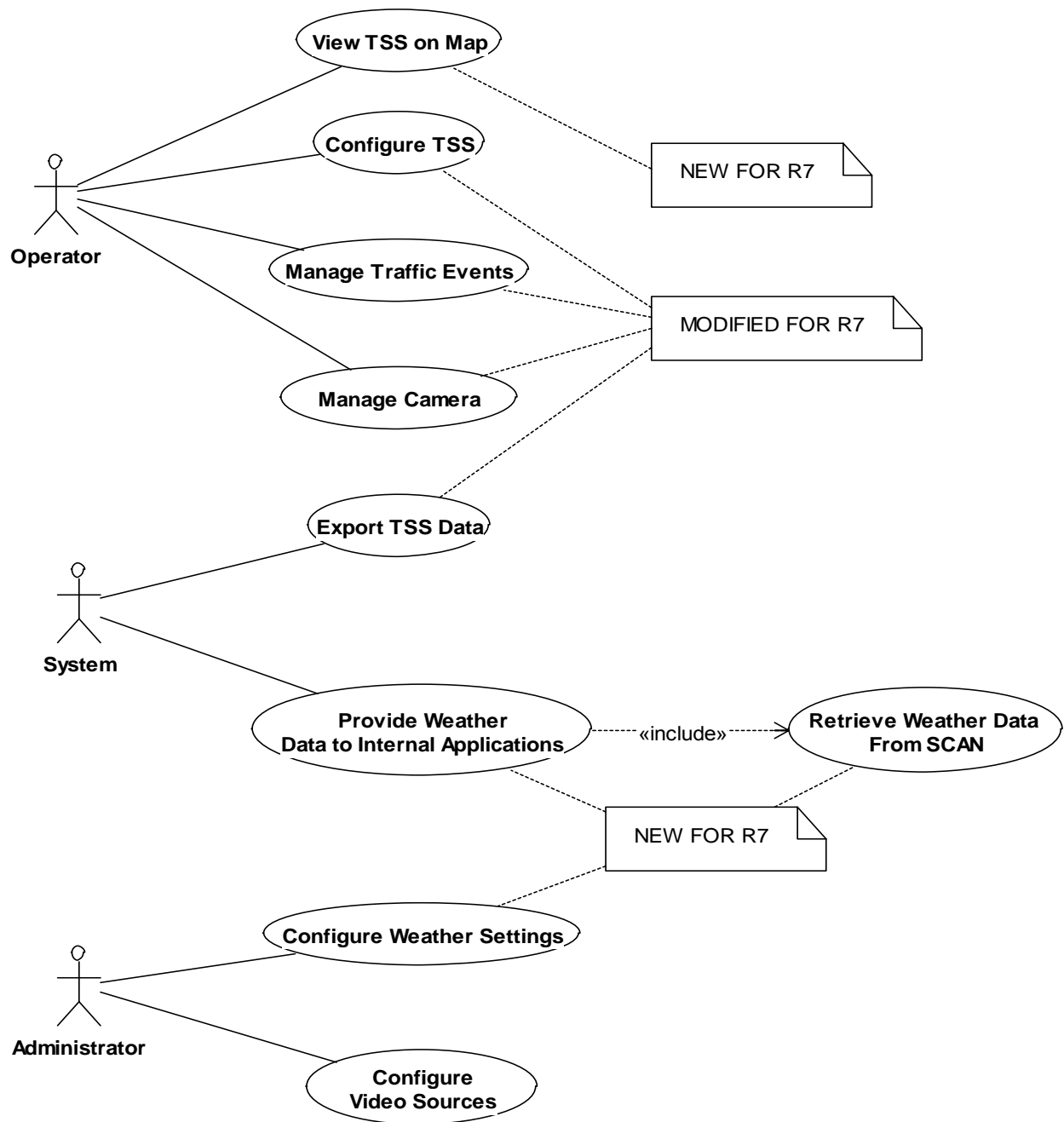
There are both CHART and Mapping use case diagrams for the Integrated Map - Detector Bearing feature.

### **4.1 CHART**

The use case diagrams depict new functionality for the CHART GUI Integrated Map and also identify existing features that will be enhanced. The use case diagrams for the Integrated Map exist in the Tau design tool in the Release7 area. The sections below indicate the title of the use case diagrams that apply to the Integrated Map.

#### **4.1.1 R7HighLevel (Use Case Diagram)**

This diagram shows the high level use cases for features added or modified as part of R7.



**Figure 4-1. R7HighLevel (Use Case Diagram)**

#### 4.1.1.1 Configure TSS (Use Case)

New in R7, a user can edit the map display options for a TSS. A user can specify a bearing for the TSS that is used to orient the TSS zone groups when they are displayed on the maps. A user can specify whether a zone group should be displayed on maps. If a user specifies that a zone group should be displayed on the maps, the zone group can be configured to display either in the direction of the TSS bearing or in the opposite direction (180 degrees

opposed) of the TSS bearing. A user can specify the order in which zone groups should appear on the maps. Zone groups with lower display order values will appear on the map closer to the TSS lat/lon position.

#### **4.1.1.2 Configure Video Sources (Use Case)**

The system allows an administrator with the “Configure Camera” right to configure video sources. Video Sources include generic unspecified video sources, "No Video Available" sources, fixed cameras, and controllable cameras including COHU, Vicon, and NTCIP cameras. Each video source can be configured to use multiple video sending devices.

#### **4.1.1.3 Configure Weather Settings (Use Case)**

The administrator will be able to configure the maximum distance from a roadway traffic event that a weather station can be to be included in the pre-selection of the road surface condition for a traffic event. (This is also referred to as a "cutoff radius"). The administrator will be able to configure the maximum age that a weather station data can have and still be included in the pre-selection of the road surface condition for a traffic event.

#### **4.1.1.4 Export TSS Data (Use Case)**

The system shall provide detector data to external systems. The system shall enforce granular, organization based user rights to allow the level of detail provided for a detector to be controlled. Two user rights (View Detailed VSO and View Summary VSO) will be used to determine if a detector's detailed volume, speed, and occupancy (VSO) data is exported, only a speed range, or no VSO data. When VSO data is provided for a detector, it will include the data for zone groups and for each zone within the group. New for R7, CHART will export map display options for each detector including: bearing, zone group display direction, and zone group display order. The detector data will be provided using the TMDD standard, with CHART extensions as needed. External systems can obtain an inventory and status of all CHART system detectors, or the ones that have changed in a certain lookback time period. They can obtain updates to the detector data (including the status) periodically with on-demand request or by subscribing to receive updates at a specified web service URL.

#### **4.1.1.5 Manage Camera (Use Case)**

An operator with the correct functional rights may perform basic operations on a camera. Please refer to the Manage Camera Use Case diagram for more detailed information.

#### **4.1.1.6 Manage Traffic Events (Use Case)**

This diagram models the actions that an operator may take that relate to traffic events. This includes responding to traffic events using field devices. New in R7, the system will pre-select the road surface condition based on data from the nearest weather station with recent data when the traffic event is opened. The operator can view the weather station that was used to make the automatic selection. The weather station data will be logged to the event

history log when the event is opened and again when it is closed. Just as in previous releases, the user may manually select the road surface condition while the event is open.

#### **4.1.1.7 Provide Weather Data to Internal Applications (Use Case)**

The system will provide an interface to be used by internal applications to obtain weather related data based on location.

#### **4.1.1.8 Retrieve Weather Data From SCAN (Use Case)**

The system will retrieve weather related data from SCAN for the purpose of making it available to internal applications.

#### **4.1.1.9 View TSS on Map (Use Case)**

The home page map shall include map layers to allow the user to view TSSs. TSS layers are shown on separate overlay layers with a separate layer for CHART TSSs and separate layers for TSSs from each external agency. The TSS layers should be below all other device layers and above the Exits/Mileposts layer. Any TSS that has a defined point location can be viewed on the home page map by clicking a link on the details page for the device.

## 4.1.2 MapAndGISUses (Use Case Diagram)

This diagram shows the mapping and GIS related use cases supported by the system.

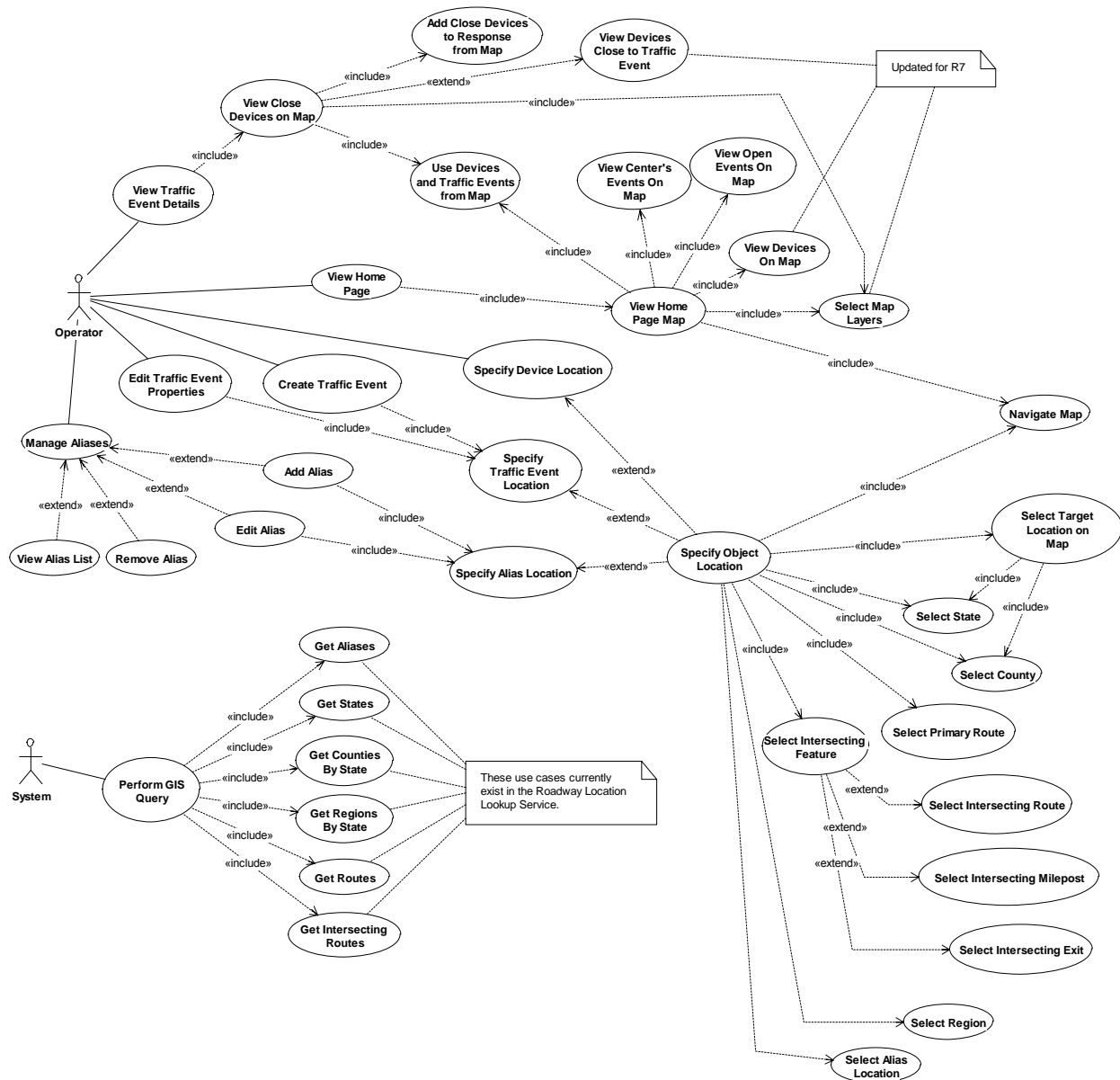


Figure 4-2. MapAndGISUses (Use Case Diagram)

### 4.1.2.1 Add Alias (Use Case)

A suitably privileged user may add a new location alias to the system. A user may specify the internal name, public name, and location of the alias.

#### **4.1.2.2 Add Close Devices to Response from Map (Use Case)**

A user may select one or more devices on the close device map for a traffic event and add them to the response plan of the target traffic event. The close devices map will update the markers of the added devices to indicate that they are in the response plan.

#### **4.1.2.3 Create Traffic Event (Use Case)**

The user with the correct functional rights may add a new traffic event. When creating a traffic event, the system will show the user a list of existing traffic events that may be duplicates of the new event being created based on the user's selections for the new event's location. External and pending events do not appear as possible duplicate events.

#### **4.1.2.4 Edit Alias (Use Case)**

A suitably privileged user may edit the information for an existing location alias. A user may specify the internal name, public name, and location of the alias.

#### **4.1.2.5 Edit Traffic Event Properties (Use Case)**

A user with sufficient privileges may alter the properties of a traffic event from the traffic event details page.

#### **4.1.2.6 Get Aliases (Use Case)**

A client application may query the list of location aliases.

#### **4.1.2.7 Get Counties By State (Use Case)**

A client application may query the complete list of counties in a specified state. The returned county data may optionally include extents and boundary information.

#### **4.1.2.8 Get Intersecting Routes (Use Case)**

A client application may query the set of routes that intersect a specified route in a particular state and county. The system will only return multiple points where the two routes intersect if those points are greater than a configurable distance from one another.

#### **4.1.2.9 Get Regions By State (Use Case)**

A client application may query the set of regions defined for a particular state.

#### **4.1.2.10 Get Routes (Use Case)**

A client application may query the list of routes defined within a particular state and county.

#### **4.1.2.11 Get States (Use Case)**

A client application may query the complete list of states. Each returned state may include optional extents data.

#### **4.1.2.12 Manage Aliases (Use Case)**

A user with sufficient privileges may use the system to manage location aliases. Refer to extending use cases for details.

#### **4.1.2.13 Navigate Map (Use Case)**

The user shall be able to pan and zoom a map display. This may include various associated features, such as pan using arrows or mouse drag, zoom using clicking, direct zoom to a specific level, draw new bounding box, etc. The pan/zoom capability provided will be dependent on the underlying map viewer chosen ... with the basic requirement being some method of pan/zoom. Other map navigation features include a Refresh Now button that immediately requests fresh data for the map view and a Close All Popups button that will close all open object callouts (popup overlays).

#### **4.1.2.14 Perform GIS Query (Use Case)**

A client application may utilize REST web services to query GIS information from the system.

#### **4.1.2.15 Remove Alias (Use Case)**

A suitably privileged user may remove a location alias from the system. Removing the location alias will not impact the location of devices and traffic events that were located using the alias in the past. Each devices or traffic event gets a copy of the alias location information when it is created.

#### **4.1.2.16 Select Alias Location (Use Case)**

A user may select from a previously defined set of alias locations in order to specify the location of an object. Doing so will populate the location with all data that was specified when the alias was created. In this case the location description will also include the public name of the alias in parentheses.

#### **4.1.2.17 Select County (Use Case)**

When specifying an object's location, a user may select a county from a list of known counties. The system will also allow the user to select a county from a list of recently used counties, or enter a free text county if the state is not Maryland. If the specified county has known extents, the map will zoom to the extent of the selected county.

#### **4.1.2.18 Select Intersecting Feature (Use Case)**

A user may select an intersecting feature along a primary route when defining the location of an object. The user may specify that the location is at, past, prior to, east of, west of, north of, or south of the selected feature.

#### **4.1.2.19 Select Intersecting Exit (Use Case)**

A user may select an intersecting exit along a primary route when setting the location of an object. If the specified exit is from the list of known exits, it will also populate the exit suffix (if applicable) and the name of the road the exit leads to (if that data is available).

#### **4.1.2.20 Select Intersecting Milepost (Use Case)**

A user may specify a milepost location using state or county milepost numbers along a route within a state and county. The system will attempt to find a defined milepost that exactly matches the specified milepost. If it cannot, it will attempt to find the two closest surrounding mileposts on the same route within a configurable distance and calculate an approximate milepost location from them (Note that this is done using straight-line interpolation. If the roadway curves significantly, the calculated point may be off the roadway. It is expected that the user would fine-tune the location using the map if the calculated point is not satisfactorily close to the roadway). If a point can be determined the map will move to this point and a marker will be placed on the map to show the user where the location is.

#### **4.1.2.21 Select Intersecting Route (Use Case)**

The user may select an intersecting route for the object location from a list of known routes that intersect the specified primary route, or may specify a free text route name if the route needed is not in the list of known intersecting routes. The system will display intersecting routes that are determined to be ramps in a separate list. If the selected intersecting route has a point location, the map will move to this location and place a marker to show the user the exact location.

#### **4.1.2.22 Select Map Layers (Use Case)**

The user shall be able to choose the map layers to be displayed on a map from a list of available layers that pertain to the map being viewed. Each device type is shown on a separate device type specific overlay layer. TSS device layers are shown on separate overlay layers with a separate layer for CHART TSSs and separate layers for TSSs from each external agency. Traffic events are shown on separate overlay layers that are specific to the traffic event type. All maps can be displayed over a pre-determined ESRI base map. The CHART Intranet base map is such a map and will be an option for the base map layer (note: the Intranet base map will be the only base map available in R5). When using the CHART base map, users may also view maps and exits as separate overlay map layers that are only visible when zoomed in on a detailed area but can be made invisible if desired.



#### **4.1.2.23 Select Primary Route (Use Case)**

The user may select a primary route for the object location from a list of known routes or may specify a free text route name if no known routes are available. The route is selected by first selecting a route type, then specifying the desired route of the specified type. If the user has selected a known route, they may specify whether the route number or route name is used. The user may also select a route direction from a pre-determined list of directions.

#### **4.1.2.24 Select Region (Use Case)**

The user may specify a region that the object is located in. If the state is MD the system will provide the user with a defined list of regions to choose from. If the state is not MD the user may enter free form text in the region field. Selecting a region will automatically cause the system to not have a county selected.

#### **4.1.2.25 Select State (Use Case)**

When specifying an object's location the user may select a state from a list of known states. Maryland will be selected by default. When this is done, the object location map will move to the extent of the specified state if it is known.

#### **4.1.2.26 Select Target Location on Map (Use Case)**

A user may click on a map in a specified way to indicate that the point they are clicking on is the desired location. When this is done during object location editing, the system will attempt to populate the state and county of the clicked location and populate those fields for the user. If the user has previously entered data about a location such as county, primary route, and intersecting feature and the new location that has been clicked is within a configurable distance of the previous location the form entries will not be lost. If, however, the new location is too far away from the previous location the user will be prompted and allowed to specify if the old location selections apply to the new point.

#### **4.1.2.27 Specify Traffic Event Location (Use Case)**

When adding a new traffic event or editing the details of an existing traffic event, a user may specify the location of the traffic event. Refer to the SpecifyObjectLocation use case for details.

#### **4.1.2.28 Specify Alias Location (Use Case)**

When adding a new location alias or editing the details of an existing location alias, a user may specify the location of the alias. Refer to the SpecifyObjectLocation use case for details.

#### **4.1.2.29 Specify Device Location (Use Case)**

A user with sufficient privileges may specify the location of the equipment. See Specify Object Location use case for details.

#### **4.1.2.30 Specify Object Location (Use Case)**

A user may specify the location of an object with the aid of a system map. This process can involve selecting a state, county or region, primary route, and intersecting feature. The system will suggest a location description based on the selections the user makes. The user may override the suggested location description if desired, but will be warned when doing so and again before submitting the location form. During the process of setting an object location, a user may press a button to reset the form. Doing this will cause all previously entered location data to be lost and the location marker(s) will be cleared from the map.

#### **4.1.2.31 Use Devices and Traffic Events from Map (Use Case)**

A user may click on an object marker on the map to use the device or traffic event that the marker represents. Doing so will open a callout (popup overlay on map) that contains information about the clicked object and HTML links that allow the user to use the device or traffic event. Clicking at a point that has multiple object markers overlapping will result in an intermediate popup that allows the user to select which of the hit markers they would like to use. Refer to the MapDeviceAndTrafficEventUses use case diagram for details.

#### **4.1.2.32 View Alias List (Use Case)**

A suitably privileged user may view the list of location aliases in the system. The detailed data for each location alias in the list shall include the Internal Name, Public Name, Location Description, County, Route, and Direction. A user may sort the list of aliases by Internal Name, Public Name, Location Description, County, and Route. A user may filter the list of aliases by County, Route, and Direction. A user may choose the columns to display in the alias list.

#### **4.1.2.33 View Center's Events On Map (Use Case)**

The system shall allow the user to view a layer on the home page map that shows the traffic events for which the user's center is responsible. This layer shall be visible by default. When the home page map is showing only events for the user's operations center, it will also show text indicating that the data is filtered. Activating this filter will move the map to an extent that includes all events controlled by the user's center and will make the event layers visible if they were not already.

#### **4.1.2.34 View Close Devices on Map (Use Case)**

The R3B3 feature that shows devices close to a traffic event on the event details page will be extended to show those devices on a map. The map will be initially panned / zoomed such that all close devices in a selected radius are shown. The user may change the defined radius to any of a pre-defined set of values. Doing so will pan/zoom the map to the extent of the devices within the new range.

#### **4.1.2.35 View Devices Close to Traffic Event (Use Case)**

The close devices map shall include map layers to allow the user to view devices that have

a defined point location and are within a defined radius of the event. Each device type shall be presented on a different map layer. The device types available for display on the map are DMS, HAR, SHAZAM, TSS, and Camera.

#### **4.1.2.36 View Devices On Map (Use Case)**

The home page map shall include map layers to allow the user to view devices. Each device type shall be presented on a different map layer. The device types available for display on the map in release 7 are DMS, HAR, SHAZAM, TSS, and Camera. Any device of these types that has a defined point location can be viewed on the home page map by clicking a link on the details page for the device.

#### **4.1.2.37 View Home Page (Use Case)**

This use case shows the home page being shown to the user. A user is shown the home page when logging in, and can view the home page at any other time during the login session. The home page always shows the user a visual indication of the number of open alerts in the new state and allows the user to quickly view summary information about the new alerts (including alert type, creation time, and description) without leaving their current view.

#### **4.1.2.38 View Home Page Map (Use Case)**

A user may view a map on the Home Page. The home page map will show all open events for the user's operating center by default.

#### **4.1.2.39 View Open Events On Map (Use Case)**

A user may choose to show all open events on the home page map. When this action is performed, the traffic event layers are made visible if they were previously hidden, and the map is zoomed to the extent of all open traffic events. Any open traffic event that has a defined point location can be viewed on the home page map by clicking a link on the details page for the traffic event.

#### **4.1.2.40 View Traffic Event Details (Use Case)**

A user with appropriate rights shall be permitted to view the details for an event. This feature is being enhanced in release 5 to include a map view that shows nearby devices on the traffic event details page.

### 4.1.3 MapDeviceAndTrafficEventUses (Use Case Diagram)

This diagram shows the ways that users may interact with devices and traffic events from the map.

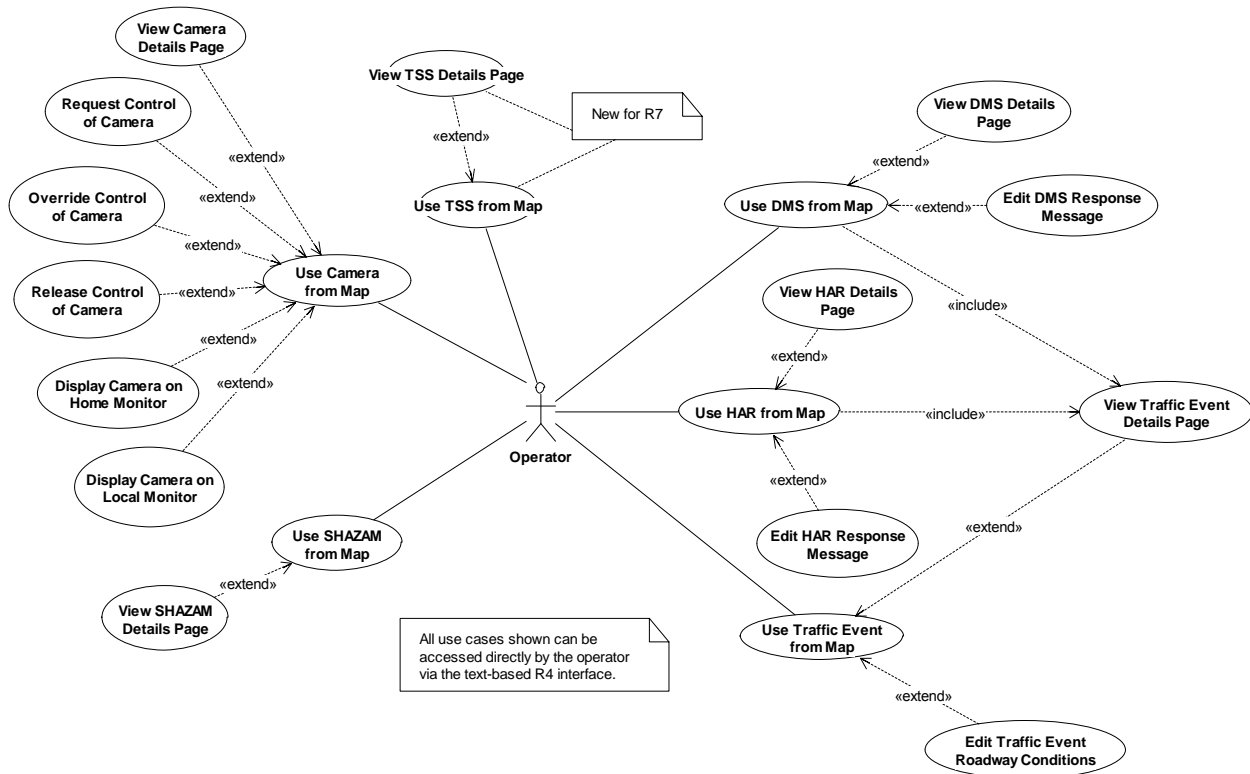


Figure 4-3. MapDeviceAndTrafficEventUses (Use Case Diagram)

#### 4.1.3.1 Display Camera on Home Monitor (Use Case)

A user may click on a link in the camera map popup to display the camera on the user's home monitor, if a home monitor has been assigned.

#### 4.1.3.2 Display Camera on Local Monitor (Use Case)

A user may click on a link in the camera map popup to display the camera on a monitor in the user's local monitor group (if a group has been assigned) and bring up the list of local monitors in the working window.

#### 4.1.3.3 Edit DMS Response Message (Use Case)

A suitably privileged user may edit and execute a DMS response message from the DMS map popup. The user may invoke either the DMS Response Message (Auto) Editor or the DMS Response Message (Manual) Editor from the DMS map popup. The edited DMS

response message will be executed when the editor form is submitted.

#### **4.1.3.4 Edit HAR Response Message (Use Case)**

A suitably privileged user may edit and execute a HAR response message from the HAR map popup. The user may invoke the HAR Response Message Editor from the HAR map popup, and the edited HAR response message will be executed when the editor form is submitted.

#### **4.1.3.5 Edit Traffic Event Roadway Conditions (Use Case)**

A user may click on a link to invoke the roadway conditions editor from the traffic event map popup.

#### **4.1.3.6 Override Control of Camera (Use Case)**

A suitably privileged user may override control of a controllable camera (that is currently being controlled by another user) from the camera map popup.

#### **4.1.3.7 Release Control of Camera (Use Case)**

A suitably privileged user may release control of a camera that is currently being controlled, from the camera map popup.

#### **4.1.3.8 Request Control of Camera (Use Case)**

A suitably privileged user may request control of a controllable camera from the camera map popup.

#### **4.1.3.9 Use Camera from Map (Use Case)**

A Camera will have a different icon on the Map depending on its operational status (e.g. online or offline). When a user causes the mouse cursor to hover over a Camera icon in the map, the name or location of the camera (as specified in the system profile) will appear. A user may click on a camera icon in the map to display summary information in a popup. The Camera map popup will display the name or location of the Camera (as configured in the system profile) and the name and operations center of the user controlling a camera, if the camera has a control session open.

#### **4.1.3.10 Use DMS from Map (Use Case)**

A DMS will have a different icon on the Map depending on its mode (e.g. online, offline, or maintenance) and whether it is currently displaying a message. When a user causes the mouse cursor to hover over a DMS icon in the map, the name of the DMS and a plain text representation of the DMS message will appear. A user may click on a DMS in the map to display summary information in a popup. The DMS map popup will display the name and location of the DMS, a representation of the DMS's current message, a list of open traffic events that currently have the DMS in their response plans, and an indicator of whether a

traffic event owns a message that is active on the DMS's message queue.

#### **4.1.3.11 Use HAR from Map (Use Case)**

A HAR will have a different icon on the Map depending on its mode (e.g. online, offline, or maintenance) and whether it is currently playing a non-default message. When a user causes the mouse cursor to hover over a HAR icon in the map, the name of the HAR will appear. A user may click on a HAR icon in the map to display summary information in a popup. The HAR map popup will display the name of the HAR, a representation of the HAR's current message, a list of open traffic events that currently have the HAR in their response plans, and an indicator as to whether a traffic event owns a message that is active on the HAR's message queue.

#### **4.1.3.12 Use SHAZAM from Map (Use Case)**

A SHAZAM will have a different icon on the Map depending on its mode (e.g. online, offline, or maintenance) and whether it has its beacons on. When a user causes the mouse cursor to hover over a SHAZAM icon in the map, the name of the SHAZAM and the current beacon state will appear. A user may click on a SHAZAM icon in the map to display the name of the SHAZAM in a popup.

#### **4.1.3.13 Use Traffic Event from Map (Use Case)**

A traffic event will have a different icon on the Map depending on the type of incident. A user may click on a traffic event icon in the map to display summary information in a popup. When a user causes the mouse cursor to hover over a traffic event icon in the map, the name of the traffic event and a description of the lane closures (if the traffic event has a defined roadway configuration) will appear. The traffic event map popup will display the name of the traffic event and a graphical representation of the lane closures (if the traffic event has a defined roadway configuration).

#### **4.1.3.14 Use TSS from Map (Use Case)**

A TSS will have a different icon on the Map depending on its mode (e.g. online, offline, or maintenance). If a TSS is online (and is not comm. marginal, comm. failure, or hardware failure) and has at least one zone group that is displayable on maps, an arrow will appear on the map for each zone group that is displayable on maps. The arrow for each zone group shall be red if the speed for that zone group is 0-30 mph, orange if the speed for that zone group is > 30 and <= 50 mph, green if speed is > 50 mph. The arrow for a zone group shall be gray if the speed data for the detector is more than 10 minutes old. The zone group arrows will be positioned on the map based on the configured zone group display order per direction. Starting at the location of the TSS, zone groups with a lower display order will appear first and zone groups with higher display orders will appear further away from the TSS lat/lon position. When a user causes the mouse cursor to hover over a TSS icon on the map, the name of the TSS and its direction will appear. A user may click on a TSS icon on the map to display the name of the TSS and zone group information in a popup. If the user has the View Detailed VSO right, the zone group information will include the name, speed,

volume, and occupancy for each zone group that is displayable on maps. If the user has the View Summary VSO right, the popup will include speed summary information for each zone group that is displayable on maps. If a user does not have either the View Detailed VSO right or the View Summary VSO right, the zone group speed data will be restricted.

#### **4.1.3.15 View Camera Details Page (Use Case)**

A user may click on a link in the Camera map popup to invoke the Camera details page in the working window.

#### **4.1.3.16 View DMS Details Page (Use Case)**

A user may click on a link in the DMS map popup to invoke the DMS details page in the working window.

#### **4.1.3.17 View HAR Details Page (Use Case)**

A user may click on a link in the HAR map popup to invoke the HAR details page in the working window.

#### **4.1.3.18 View SHAZAM Details Page (Use Case)**

A user may click on a link in the SHAZAM map popup to invoke the SHAZAM details page in the working window.

#### **4.1.3.19 View Traffic Event Details Page (Use Case)**

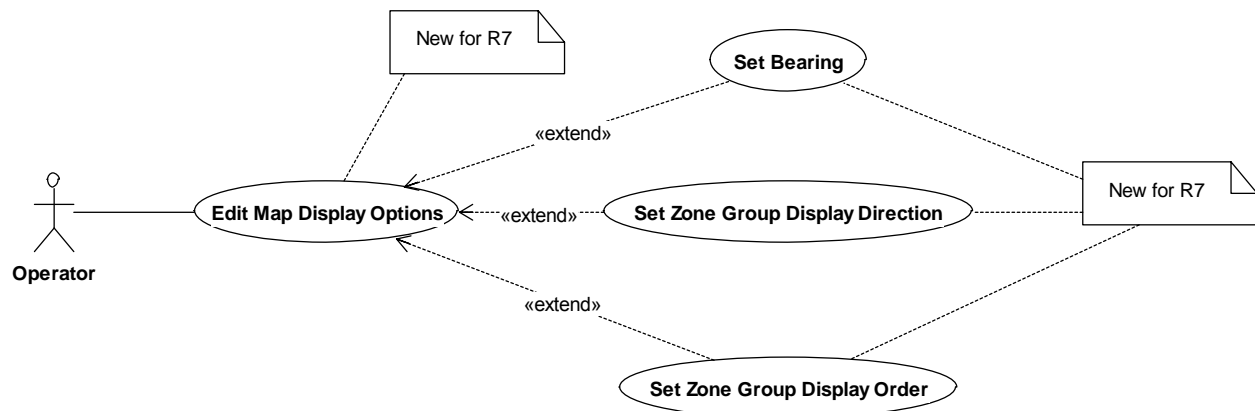
A user may click on a) a link in the traffic event map popup, b) a traffic event listed in the DMS map popup, or c) a traffic event listed in the HAR map popup to invoke the traffic event details page in the working window.

#### **4.1.3.20 View TSS Details Page (Use Case)**

A user may click on a link in the TSS popup on the map to invoke the TSS details page in the working window.

#### 4.1.4 ConfigureTSS (Use Case Diagram)

This Use Case Diagram identifies the new actions for R7 which can be performed to configure the "Map Display Options" for a TSS. This is an expansion of the action "Configure TSS" in the "R7 High Level" Use Case Diagram.



**Figure 4-4. ConfigureTSS (Use Case Diagram)**

##### 4.1.4.1 Edit Map Display Options (Use Case)

New for R7, a user with the configure TSS functional right can edit the map display properties for any TSS that has a defined location (lat/lon). The map display options can be edited with the TSS in any mode (online, offline or maintenance mode). The map display properties include: TSS bearing, zone group display direction, and zone group display order.

##### 4.1.4.2 Set Bearing (Use Case)

New for R7, a user can specify the bearing for any TSS that has a defined location (lat/lon). A bearing of 0 degrees shall mean the bearing is due East. The bearing shall grow counter-clockwise such that a bearing of 90 indicates due North, 180 indicates due West, and 270 indicates due South. A TSS will not have a defined bearing when it is initially created. The TSS bearing is used to orient the zone groups for the TSS on the maps.

##### 4.1.4.3 Set Zone Group Display Direction (Use Case)

New for R7, a user can specify the display direction for each zone group. The display direction will indicate how the zone group should be displayed on maps. A user can specify that a zone group should either be displayed on maps or not be displayed on maps. For zone groups that are displayed on maps, a user can specify whether they are displayed using an arrow that points in the direction of the TSS bearing or using an arrow that points in the opposite direction (180 degrees opposed) of the TSS bearing. When a user changes a zone group from not displayable on maps to displayable on maps, the system shall display a



warning that the zone group name may be displayed on the Internet map. A zone group will be set to not displayable on maps when it is initially created.

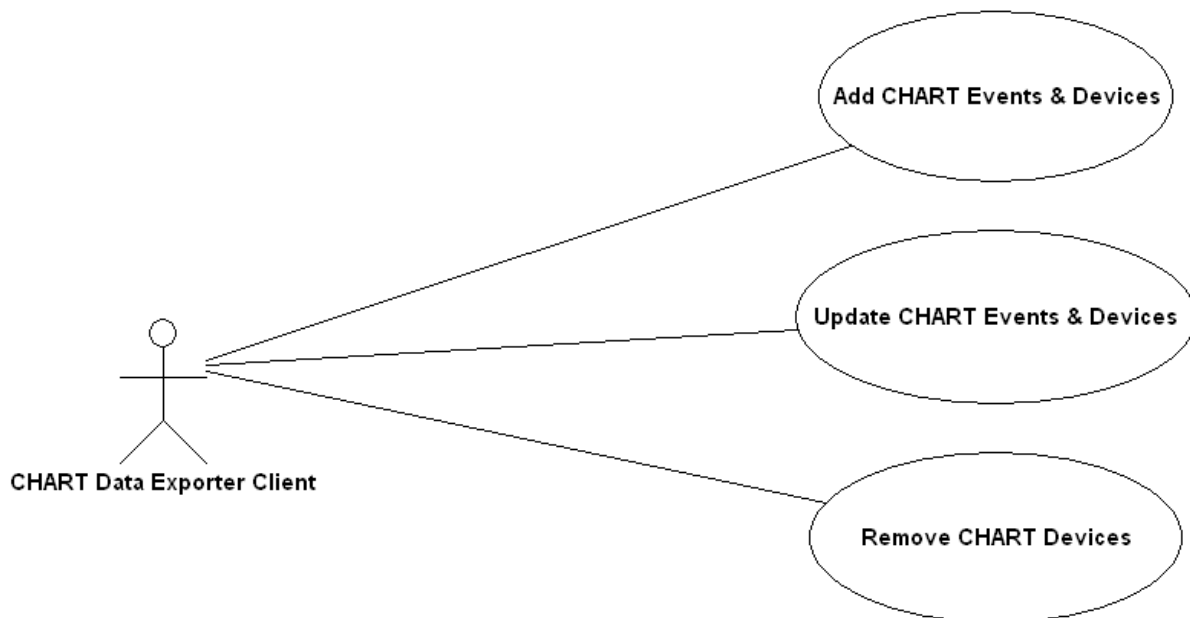
#### 4.1.4.4 Set Zone Group Display Order (Use Case)

New for R7, a user can specify the display order for each zone group relative to other zone groups of the TSS with the same display bearing. Starting at the location of the TSS, zone groups with a lower display order will appear first on the map and zone groups with higher display orders will appear further away from the TSS lat/lon position.

## 4.2 Mapping

The use case diagrams depict new functionality for the CHART Intranet and Internet Mapping Application and also identify existing features that will be accessible via the CHART Intranet Mapping Application. The sections below indicate the title of the use case diagrams that apply to the CHART Intranet and Internet Mapping Application.

### 4.2.1 Data Exporter Synchronization



**Figure 4-5. Data Exporter Synchronization (Use Case Diagram)**

#### 4.2.1.1 Synchronize Add Events & Devices (Use Case)

The Synchronization Application shall listen to the CHART Data Exporter Client's request when an add event occurred. In Mapping R5, the Synchronization Application shall

synchronize add events when a new CHART Event, or Device (DMS, HAR, SHAZAM, CAMERA, and DETECTOR) is added in CHART. The Synchronization Application shall also synchronize any new CHART Events, or Devices (DMS, HAR, SHAZAM, CAMERA, DETECTOR) when a full inventory update occurs in CHART Data Exporter Client.

#### **4.2.1.2 Synchronize Update Events & Devices (Use Case)**

The Synchronization Application shall listen to the CHART Data Exporter Client's request when an update event occurred. In Mapping R5, the Synchronization Application shall synchronize update events when an existing CHART Event or Device (DMS, HAR, SHAZAM, CAMERA, and DETECTOR) is updated in CHART. The Synchronization Application shall also synchronize any updates of CHART Events, or Devices (DMS, HAR, SHAZAM, CAMERA, DETECTOR) when a full inventory update occurs in CHART Data Exporter Client.

#### **4.2.1.3 Synchronize Remove Devices (Use Case)**

The Synchronization Application shall listen to the CHART Data Exporter Client's request when a remove event occurred. In R5, the Synchronization Application shall synchronize remove events when an existing CHART Device (DMS, HAR, SHAZAM, CAMERA, and DETECTOR) is removed in CHART. The Synchronization Application shall also synchronize any removal of CHART Devices (DMS, HAR, SHAZAM, CAMERA, DETECTOR) when a full inventory update occurs in CHART Data Exporter Client. The removal of CHART Events is handled by a nightly schedule job.

## 4.2.2 DisplayTSS (Use Case Diagram)

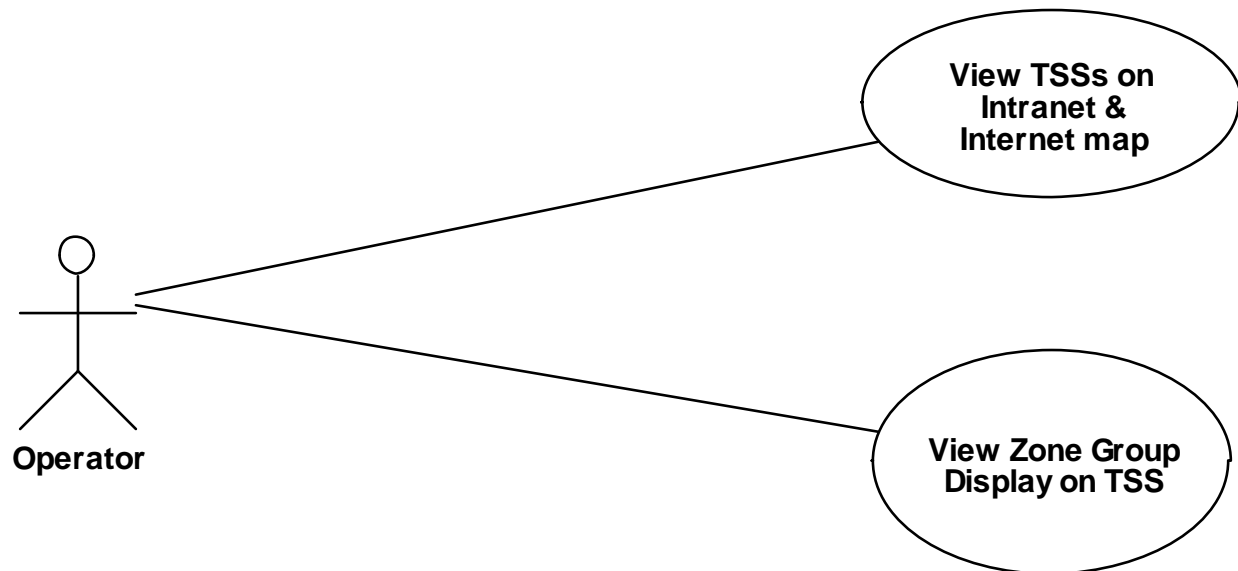


Figure 4-6 DisplayTSS (Use Case Diagram)

### 4.2.2.1 Operator (Actor)

### 4.2.2.2 View TSSs on Intranet & Internet map (Use Case)

This use case is about viewing the TSSs after they been added or updated and it has a bearing in CHART. The Map will display the Detector arrow rotated according to the bearing generated by the CHART system.

### 4.2.2.3 View Zone Group Display on TSS (Use Case)

The map will display a detector zone group only if the 'Display On Maps' indicator from the CHART system indicates to do so. If a TSS is changed to have no displayable zone groups (in CHART), it will not be displayed on the map. If a TSS has one or more displayable zone groups marked as map displayable (in CHART) the TSS will be displayed on the map.

## 5 Detailed Design – Integrated Map – Detector Bearing

---


### 5.1 Human-Machine Interface

#### TSS Configuration

This section describes forms used to configure a TSS for display on the maps. The maps and forms have been changed to allow a user to specify display properties for a TSS in order to display the TSS and its zone groups. The TSS display properties include: bearing, zone group display direction, and zone group display order. A page for setting these properties can be accessed from the TSS details page. The TSS and its zone groups, if applicable, will be displayed on the Home Page and Close Devices maps in the CHART GUI and will be exported for display on the Intranet and Internet maps.

#### TSS Map Display Options

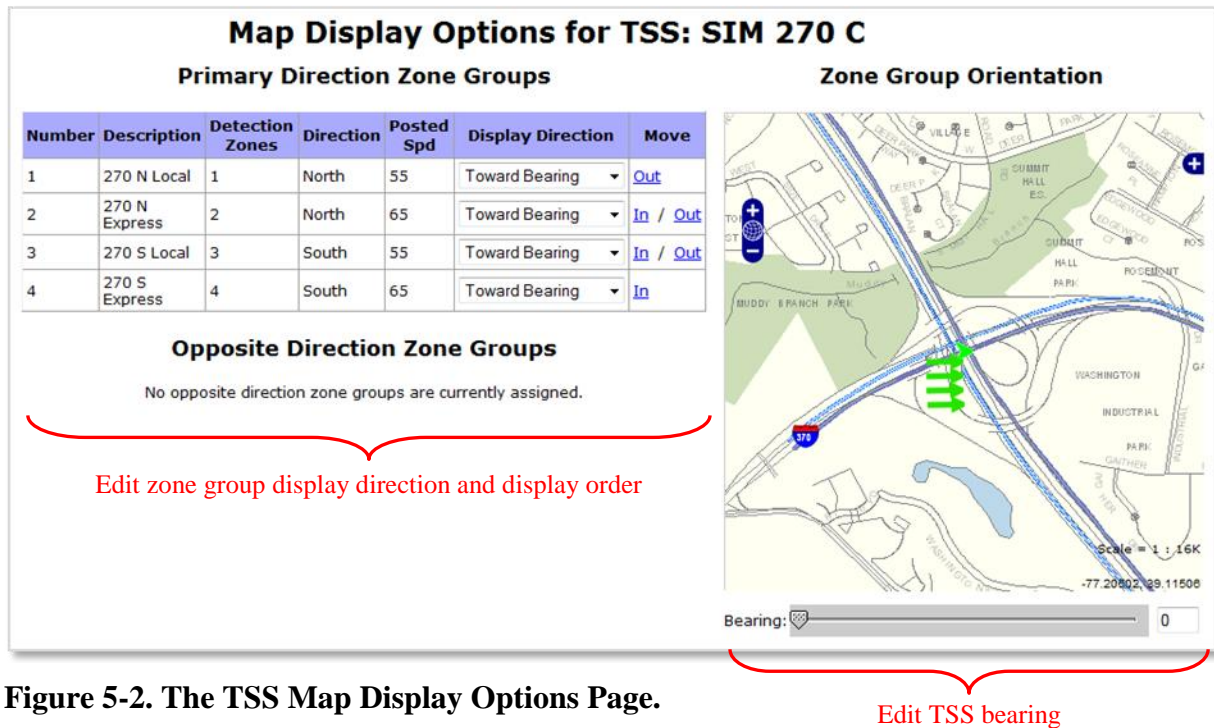
The Map Display Options for a TSS can be edited by using the Edit Map Display Options page. This page can be accessed via a link on the TSS Details page in the Zone Groups configuration section (see **Error! Reference source not found.** below). This link will be available only if the TSS has a location with a defined latitude/longitude.



Zone Groups: [\(Edit Map Display Options\)](#)

**Figure 5-1 The Edit Map Display Options Link in the Zone Groups Configuration Section**

The Edit Map Display Options page contains a form for updating the zone group display direction and display order, as well as a map that can be used to set the bearing for the TSS (see below).

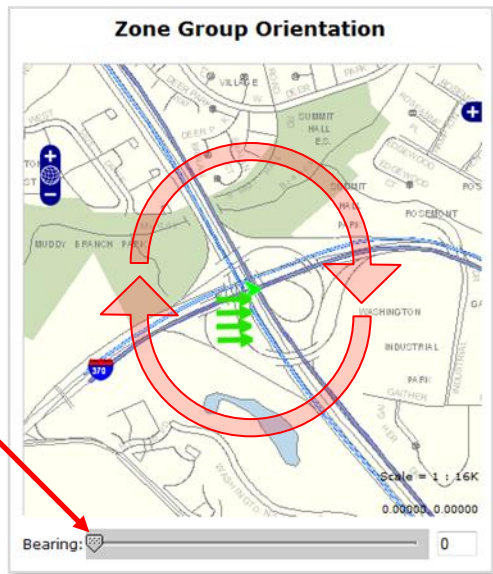


**Figure 5-2. The TSS Map Display Options Page.**

### TSS Bearing

The TSS bearing determines the orientation of zone groups on the maps. The TSS bearing has values of 0 to 359 degrees. A bearing of 0 degrees means the bearing is due East. The bearing grows counter-clockwise such that a bearing of 90 degrees indicates due North, 180 degrees indicates due West and 270 degrees indicates due South. A bearing of 359 degrees is 1 degree South of due East. When a TSS is initially created, the bearing is undefined.

The bearing can be set for any TSS that has a location with a defined latitude and longitude. The bearing can be edited either by using the slider (located below the map) or by clicking and dragging on the map in the direction of the desired bearing (see **Error! Reference source not found.** below). As the slider pointer is moved from left to right, the bearing value increases in the range from 0 to 359 degrees (and the orientation of the bearing pointer on the map is updated accordingly). Clicking and dragging on the map will move the bearing pointer in the direction of the mouse pointer (and update the slider value accordingly).



**Figure 5-3. Setting the TSS Bearing Using the Slider or the Map.**



## Zone Group Display Direction

The Display Direction of a zone group determines if and how a zone group will be displayed on the maps. The values for Display Direction include: Toward Bearing, Away from Bearing, and Do Not Display. A zone group with a Display Direction of Toward Bearing will be displayed on the maps using the same bearing as the TSS. A zone group with a Display Direction of Away from Bearing will be displayed on the maps using a bearing that is 180° opposite the bearing of the TSS (i.e. if the TSS has a bearing of 90°, the zone group will be displayed using a bearing of 270°). A zone group with a Display Direction of Do Not Display will not appear on the maps.

When a zone group is added to a TSS, the display direction is given a default value based on the direction of the TSS and the direction of the zone group. If the direction of the TSS has a value of “None”<sup>1</sup>, the display direction of the TSS will be set to “Toward Bearing” regardless of the direction of the zone group. If the direction of the TSS has a value other than None, the zone group display direction will be set to: “Toward Bearing” if the zone group direction matches the TSS direction (e.g. North and North), “Away from Bearing” if the zone group direction is opposite to the TSS direction (e.g. North and South), or “Toward Bearing” if the zone group direction and TSS direction cannot be compared (e.g. North and Inner Loop).

The zone groups are organized on the forms based on the value of the Display Direction (see Figure 5-4 below). Any Primary Direction zone groups are listed first, next any Opposite Direction zone groups are listed, and lastly any Other zone groups are listed. When the Display Direction of a zone group is updated, the zone group is moved to the appropriate group on the forms.

Primary Direction Zone Groups						
Number	Description	Detection Zones	Direction	Posted Spd	Display Direction	Move
1	270 N Local	1	North	55	Toward Bearing ▾	<a href="#">Out</a>
2	270 N Express	2	North	65	Toward Bearing ▾	<a href="#">In</a>
Opposite Direction Zone Groups						
Number	Description	Detection Zones	Direction	Posted Spd	Display Direction	Move
3	270 S Local	3	South	55	Away from Bearing ▾	
Other Zone Groups						
Number	Description	Detection Zones	Direction	Posted Spd	Display Direction	Move
4	270 S Express	4	South	65	Do Not Display ▾	

} Display toward bearing

} Display away from bearing

} Do not display on maps

**Figure 5-4. Zone Groups Organized by Display Direction.**

<sup>1</sup> Direction values for a TSS include: None, North, East, South, West, Inner Loop, Outer Loop, South/North, East/West, and Inner Loop/Outer Loop.

When a zone group that is not displayable on maps (i.e. the Display Direction is Do Not Display) is updated to be displayable on maps (i.e. the Display Direction is either Toward Bearing or Away from Bearing), the user will be warned that the zone group name may now be displayed on the Intranet and/or Internet maps (see Figure 5-6 below).

**Map Display Options for TSS: SIM 270 C**

**Primary Direction Zone Groups**

Number	Description	Detection Zones	Direction	Posted Spd	Display Direction	Move
2	270 N Express	2	North	65	Toward Bearing	Out
1	270 N Local	1	North	55	Toward Bearing	In

**Zone Group Orientation**

**Opposite Direction Zone Groups**

Number	Description	Detection Zones	Direction	Posted Spd	Display Direction	Move
4	270 S Express					
3	270 S Local	3	South	55	Away from Bearing	

The page at http://localhost:8080 says:

**Warning:** Making this zone group map displayable implies that the zone group name '270 S Local' may be displayed on the Intranet and/or Internet maps.

OK

Bearing: 118

Update Close

**Figure 5-5. Warning That a Zone Group is Being Changed to Displayable on Maps.**

### Zone Group Display Order

The Display Order of a zone group determines where a zone group will be displayed on the maps in relation to the TSS latitude/longitude. Zone group arrows will be displayed on the maps based on the configured zone group display order per direction. Starting at the location of the TSS, zone groups with lower display orders will appear first and zone groups with higher display orders will appear further away.

The Display Order of a zone group can be edited by clicking on either the “In” or “Out” link in the “Move” column of the table (see Figure 5-6 below). Clicking on the “In” link will cause the zone group to be positioned closer to the TSS, and clicking on the “Out” link will cause the zone group to be positioned further away from the TSS.



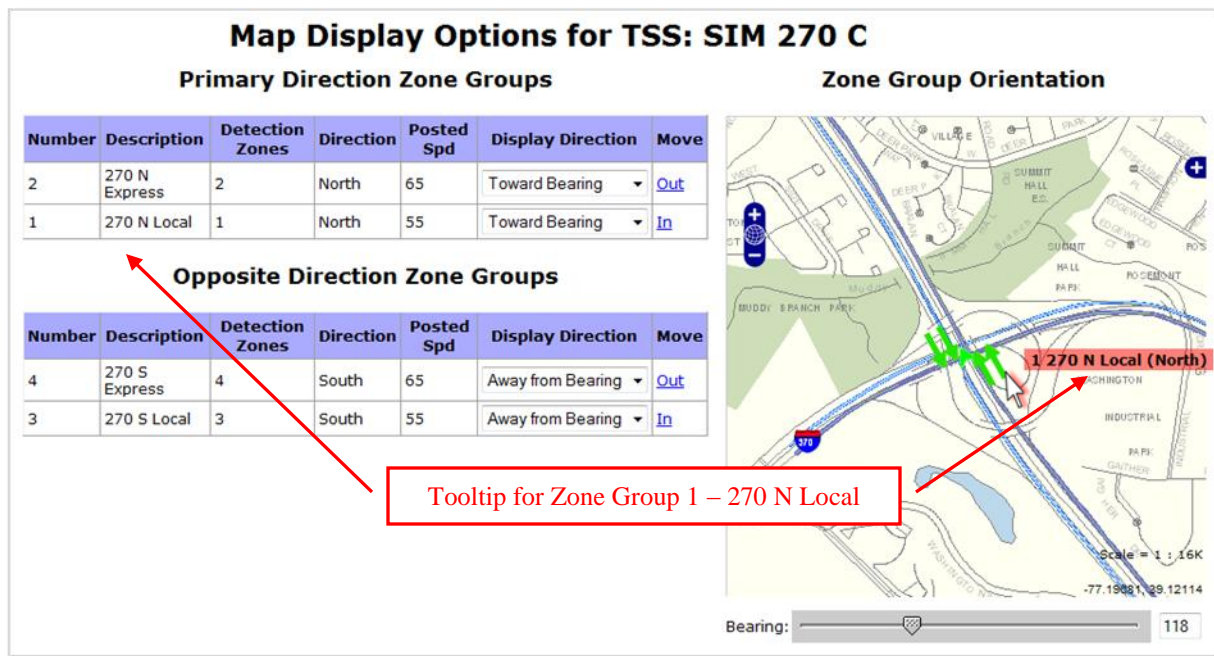
Primary Direction Zone Groups						
Number	Description	Detection Zones	Direction	Posted Spd	Display Direction	Move
2	270 N Express	2	North	65	Toward Bearing ▾	<a href="#">Out</a>
1	270 N Local	1	North	55	Toward Bearing ▾	<a href="#">In</a>

Opposite Direction Zone Groups						
Number	Description	Detection Zones	Direction	Posted Spd	Display Direction	Move
3	270 S Local	3	South	55	Away from Bearing ▾	<a href="#">Out</a>
4	270 S Express	4	South	65	Away from Bearing ▾	<a href="#">In</a>

**Figure 5-6. Updating the Zone Group Display Order by Using the Move In and Move Out Links.**

When the display order of a zone group is updated, the change is reflected on the map for the Edit Map Display Options page. The number, name and direction for each zone group can be viewed from the map by hovering the mouse pointer over the zone group arrow (see Figure 5-7 below).



**Figure 5-7. Zone Group Tooltip Showing the Number, Name, and Direction.**

## TSSs Displayed on Maps

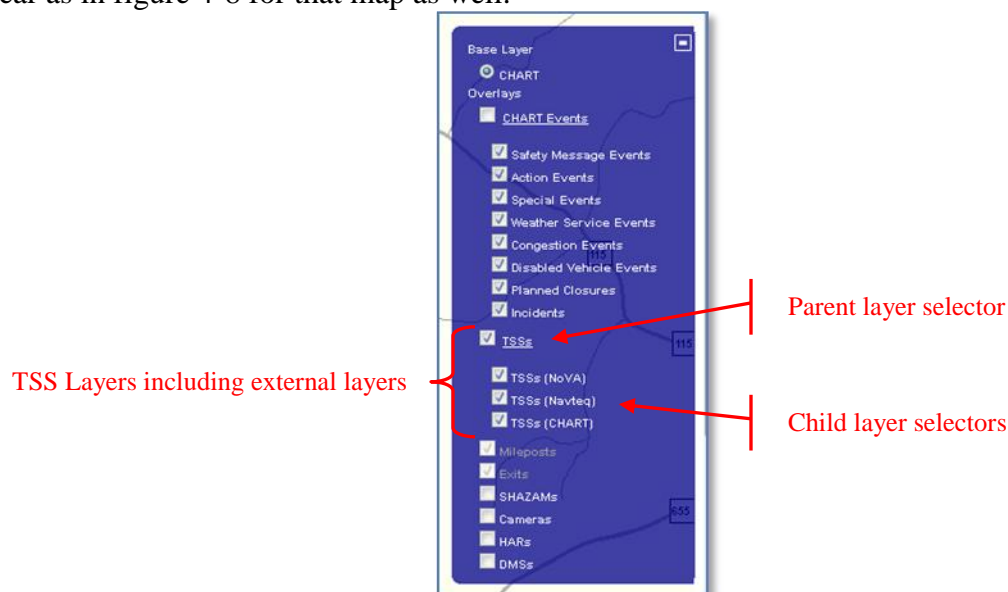
This section describes the display of TSSs on the maps within the CHART GUI.

### Layers Selection and Zoom Based Visibility

The TSS child layer selectors are grouped in the map layer selector under a parent TSSs layer selector (see Figure 5-8 below). The TSS child layer selectors include a layer selector for CHART TSSs and a separate layer selector for each external agency from which CHART has imported TSSs. Checking the selector check box for the parent TSSs layer will make any child layers that are checked visible on the map. Un-checking the selector check box for the parent TSSs layer will make all child layers invisible on the map. Clicking on the link for the parent TSSs layer selector will cause the child layer selectors to collapse/expand under the parent selector.

The TSS layers are not visible on the map at the highest zoom levels. The zoom levels at which the TSS layers are visible are configurable at the application level.

These same rules for TSS map layers apply on the nearby devices map. The TSS layers will appear as in figure 4-8 for that map as well.



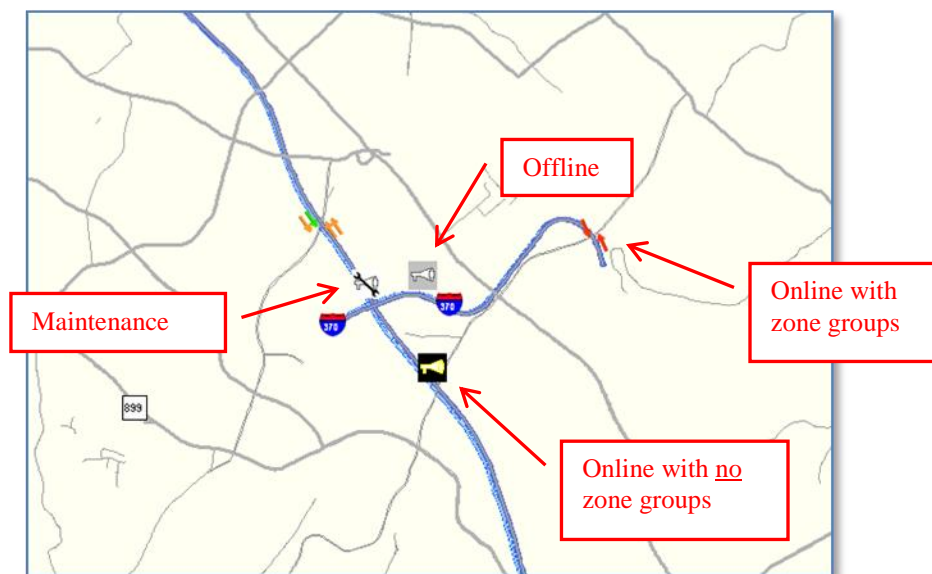
**Figure 5-8. Map Layer Selector Showing TSS Layers Selected.**

### TSS Icons

All TSSs that have a location with a defined latitude/longitude are displayed on the maps in the CHART GUI. The TSSs are displayed using either an icon from the list page or arrows to show the zone groups. A TSS that has a defined bearing, at least one defined zone group that is displayable on maps, and is online (and not comm. failed, comm. marginal, or hardware failed)

will be displayed using a colored arrow per zone group to depict current speed information for the TSS (see Figure 5-9 below). The arrow for each zone group will be red if the speed for that zone group is between 0 and 30 mph, orange if the speed for that zone group is greater than 30 mph and less than or equal to 50 mph, or green if the speed is greater than 50 mph. The arrow for a zone group will be gray if the speed data for the detector is more than 10 minutes old. A TSS that does not have a defined bearing, has no defined zone groups that are displayable on maps, is not online, or is online and comm. failed, comm. marginal, or hardware failed, will be displayed using the icon from the device list for that particular TSS.

Zone group arrows are displayed on the maps based on the configured zone group display order per direction. Starting at the location of the TSS, zone groups with lower display orders will appear first and zone groups with higher display orders will appear further away.



**Figure 5-9. Icons of TSSs in Different States on the Map.**

### TSS Tooltips and Callouts

Each TSS on the map will display a tooltip when the user hovers the mouse over the TSS icon (or zone group arrows). The tooltip will contain the name of the TSS (see Figure 5-10 below). Each TSS on the map will also display a callout when the user clicks on the TSS icon (or zone group arrows). The callout will contain: the name, the icon indicating its current mode (online, maintenance, or offline) and current status (comm. failure, comm. marginal, or hardware failure) if applicable, a link to the TSS details page, the location, and one or more tables of zone group information. The zone group information will contain configuration information and current traffic parameters for each zone group (organized by direction) including: the zone group name, the current volume, the current speed, and the current occupancy. This information will be available to any user with the detailed VSO functional right. A user with the view summary VSO functional right will see a current speed summary instead of the actual speed.



**Figure 5-10. TSS Tooltip and Callout Showing Number, Name, Direction, and Zone Groups.**

## **5.2 System Interfaces**

### **5.2.1 Class Diagrams**

#### **5.2.1.1 TSSManagement (Class Diagram)**

This class diagram contains the interfaces, structs, and typedefs that are to be defined in IDL and provide the external interface to the TSSManagement package of the CHART II system.

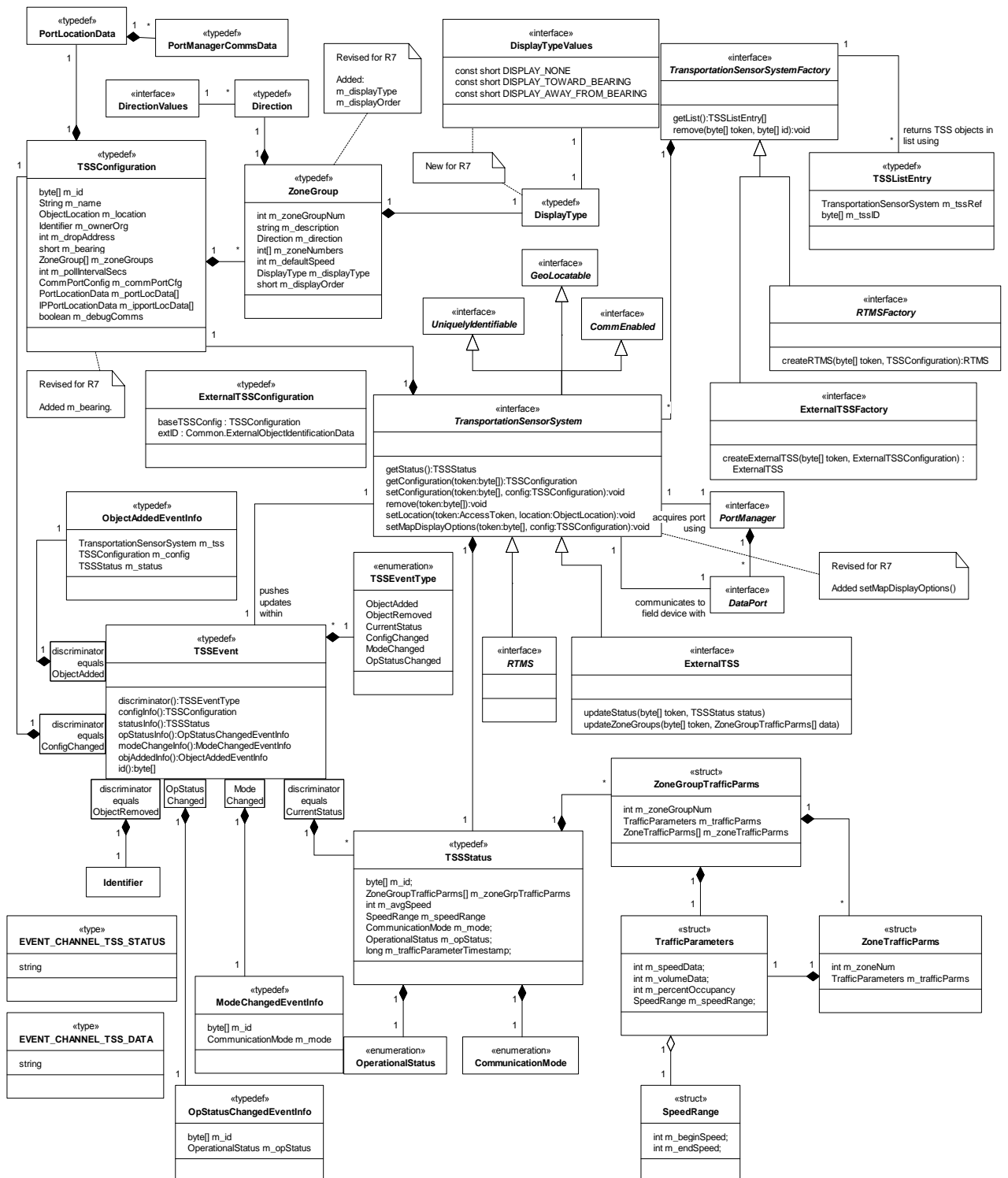


Figure 5-11 TSSManagement (Class Diagram)

#### **5.2.1.1.1 CommEnabled (Class)**

The CommEnabled interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by TrafficEvents within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by TrafficEvents), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

#### **5.2.1.1.2 CommunicationMode (Class)**

The CommunicationMode class enumerates the modes of operation for a device: ONLINE, OFFLINE, and MAINT\_MODE. ONLINE is used to indicate the device is available to the operational system. OFFLINE is used to indicate the device is not available to the online system and communications to the device have been disabled. MAINT\_MODE is used to indicate that the device is available only for maintenance / repair activities and testing.

#### **5.2.1.1.3 DataPort (Class)**

A DataPort is a port that allows binary data to be sent and received. Ports of this type support a receive method that allows a chunk of all available data to be received. This method prevents callers from having to issue many receive calls to parse a device response. Instead, this receive call returns all available data received within the timeout parameters. The caller can then parse the data within a local buffer. Using this mechanism, device command and response should require only one call to send and one call to receive.

#### **5.2.1.1.4 Direction (Class)**

This type defines a short value that is used to indicate a direction of travel as defined in DirectionValues.

#### **5.2.1.1.5 DirectionValues (Class)**

This interface contains constants for directions as defined in the TMDD.

#### **5.2.1.1.6 DisplayType (Class)**

This type defines a short value that is used to indicate the display type of a zone group as defined in DisplayTypeValues. The display type indicates whether a zone group arrow will appear on the maps and if so the direction the zone group arrow will be displayed in relation to the TSS bearing (e.g. toward the bearing or away from the bearing).

#### **5.2.1.1.7 DisplayTypeValues (Class)**

This interface contains constants for display types used by zone groups. The constant include: DISPLAY\_NONE (do not display on maps), DISPLAY\_TOWARD\_BEARING (display toward the TSS bearing), and DISPLAY\_AWAY\_FROM\_BEARING (display 180

degrees away from TSS bearing).

#### **5.2.1.1.8 EVENT\_CHANNEL\_TSS\_DATA (Class)**

This is a static string that contains the name of the event channel used to push events that contain Transportation Sensor System traffic parameter data. The following TSSEventTypes are pushed on EVENT\_CHANNEL\_TSS\_DATA channels:

CurrentStatus

#### **5.2.1.1.9 EVENT\_CHANNEL\_TSS\_STATUS (Class)**

This is a static string that contains the name of the event channel used to push events relating to the change in a Transportation Sensor System status and/or configuration. The following TSSEventTypes are pushed on EVENT\_CHANNEL\_TSS\_STATUS channels:

ObjectAdded

ObjectRemoved

ConfigChanged

ModeChanged

OpStatusChanged

#### **5.2.1.1.10 ExternalTSS (Class)**

This interface represents an External Systems TSS in the Chart System. I.E. a proxy for a physical TSS outside of Chart.

#### **5.2.1.1.11 ExternalTSSConfiguration (Class)**

This class holds configuration data for an ExternalTSS. It extends the TSSConfiguration data by including a reference to the base TSSConfig.

baseTSSConfig - Reference to the base TSSConfig.

extID - This objects holds the External System Name / Ext Agency / Ext id for this External TSS. This uniquely identifies it in Chart.

#### **5.2.1.1.12 ExternalTSSFactory (Class)**

This interface extends the TransportationSensorSystemFactory interface to allow support of ExternalTSS objects in Chart.



#### **5.2.1.1.13 GeoLocatable (Class)**

This interface is implemented by objects that can provide location information to their users.

#### **5.2.1.1.14 Identifier (Class)**

Wrapper class for a CHART2 identifier byte sequence. This class will be used to add identifiable objects to hash tables and perform subsequent lookup operations.

#### **5.2.1.1.15 ModeChangedEventInfo (Class)**

This struct contains information pushed with a ModeChanged event.

m\_id - The ID of the TSS whose communication mode has changed.

m\_mode - The new communication mode for the TSS.

#### **5.2.1.1.16 ObjectAddedEventInfo (Class)**

This structure contains information passed in the ObjectAdded event pushed on a TSS status event channel. It contains the object reference that has been added along with its configuration values and current status values.

#### **5.2.1.1.17 OperationalStatus (Class)**

The OperationalStatus class enumerates the types of operational status a device can have: OK (normal mode), COMM\_FAILURE (no communications to the device), or HARDWARE\_FAILURE (device is reachable but is reporting a hardware failure).

#### **5.2.1.1.18 OpStatusChangedEventInfo (Class)**

This struct contains data passed with an OpStatusChanged event.

m\_id - The ID of the TSS whose operational status has changed.

m\_opStatus - The new operational status for the device.

#### **5.2.1.1.19 PortLocationData (Class)**

This class contains configuration data that specifies the communication server(s) to use to communicate with a device.

m\_commsData - One or more objects identifying the communications server (PortManager) to use to communicate with the device, in order of preference.

m\_portType - The type of port to use to communicate with the device (ISDN modem, POTS modem, direct, etc.)

m\_portWaitTimeSecs - The maximum number of seconds to wait when attempting to acquire a port from a port manager.

#### **5.2.1.1.20 PortManager (Class)**

A PortManager is an object that manages shared access to communications port resources. The getPort method is used to request the use of a port from the PortManager. Requests for ports specify the type of port needed, the priority of the request, and the maximum time the requester is willing to wait if a port is not immediately available. When the port manager returns a port, the requester has exclusive use of the port until the requester releases the port back to the PortManager or the PortManager reclaims the port due to inactivity.

#### **5.2.1.1.21 PortManagerCommsData (Class)**

This class contains values that identify a port manager and the phone number to dial to access a device from the given port manager. This class exists to allow for the phone number used to access a device to differ based on the port manager to take into account the physical location of the port manager within the telephone network. For example, when dialing a device from one location the call may be long distance but when dialing from another location the call may be local.

#### **5.2.1.1.22 RTMS (Class)**

The Remote Traffic Microwave Sensor (RTMS) is a detector manufactured by EIS, Inc. capable of providing lane level volume, speed, and occupancy data for up to 8 lanes of a roadway at a single location. This interface serves to identify TransportationSensorSystem objects as being of the type RTMS. It also provides a place holder for future operations that may not apply to TSS objects in general and are instead RTMS specific.

#### **5.2.1.1.23 RTMSFactory (Class)**

Objects which implement RTMSFactory are capable of adding an RTMS to the system.

#### **5.2.1.1.24 SpeedRange (Class)**

This struct is used to specify a speed range. The speed range is defined in MPH and has an upper and lower limit inclusive. Note: m\_endSpeed of zero means range is > m\_beginSpeed. MPH is implied.

#### **5.2.1.1.25 TrafficParameters (Class)**

This struct contains traffic parameters that are sensed and reported by a Traffic Sensor System such as the RTMS.

m\_speedData - The arithmetic mean of the speeds collected over a sample period in miles per hour in tenths. (thus 550 == 55.0 MPH) Valid values are 0 to 2550. A value of 65535 is used to indicate a missing or invalid value (such as when the volume for the sample period is zero).

m\_volumeData - The count of vehicles for the sample period. Valid values 0 to 65535. A value of 65535 represents a missing value.

m\_percentOccupancy - The percentage of occupancy of the roadway in tenths of a percent. (thus 1000 = 100.0 percent). Valid values are 0 to 1000. A value of 65535 represents a missing or invalid value.

#### **5.2.1.1.26 TransportationSensorSystem (Class)**

A Transportation Sensor System (TSS) is a generic term used to describe a class of technology used for detection within the transportation industry. Examples of TSS devices range from the advanced devices, such as RTMS, to basic devices, such as single loop detectors.

This software interface is implemented by objects that provide access to the traffic parameters sensed by a Transportation Sensor System. Transportation Sensor Systems are capable of providing detection for one or more detection zones. A single loop detector would have one detection zone, while an RTMS would have 8 detection zones.

#### **5.2.1.1.27 TransportationSensorSystemFactory (Class)**

This interface is implemented by objects that are used to create and serve TransportationSensorSystem (TSS) Objects. All factories of TSS objects can return the list of TSS objects which they have created and serve. Derived interfaces are used to provide factories to create specific make, models, and types of TransportationSensorSystem objects.

#### **5.2.1.1.28 TSSConfiguration (Class)**

This class holds configuration data for a transportation sensor system (TSS) as follows:

m\_id - The unique identifier for this TSS. This field is ignored when the object is passed to the TSS to change its configuration.

m\_name - The name used to identify the TSS.

m\_location - A descriptive location of the TSS.

m\_dropAddress - The drop address for the device.

m\_bearing - The bearing in degrees for displaying the TSS on the map. Valid values are from -1 to 359 (-1 = bearing not defined, 0 = East, 90 = North, 180 = West, and 270 = South). The default value is -1.

m\_zoneGroups - Logical groupings of detection zones, used to provide a single set of traffic parameters for one or more detection zones.

m\_pollIntervalSecs - The interval on which the TSS should be polled for its current traffic parameters (in seconds).

m\_commPortCfg - Communication configuration values.

m\_portLocData - Configuration information that determines which port manager(s) should be used to establish a connection with the SensorSystem.

m\_debugComms - Flag used to enable/disable the logging of communications data for this TSS. When enabled, command and response packets exchanged with the device are logged to a debugging log file.

#### **5.2.1.1.29 TSSEvent (Class)**

This class is a CORBA union that contains varying data depending on the current value of the discriminator.

If the discriminator is ConfigChanged, this union contains a TSSConfig object.

If the discriminator is ObjectAdded, this union contains an ObjectAddedEventInfo object.

If the discriminator is ObjectRemoved, this union contains a byte[] containing the unique identifier for the Traffic Sensor System that was removed.

If the discriminator is CurrentStatus the union contains an array of one or more TSSStatus objects.

If the discriminator is ModeChanged, the union contains a ModeChangedEventInfo.

If the discriminator is OpStatusChanged, the union contains an OpStatusChangedEventInfo object.

#### **5.2.1.1.30 TSSEventType (Class)**

This enumeration defines the types of events that may be pushed on an event channel by a Transportation Sensor Status object. The values in this enumeration are used as the discriminator in the TSSEvent union.

ObjectAdded - a TransportationSensorSystem has been added to the system.

ObjectRemoved - a TransportationSensorSystem has been removed from the system.

CurrentStatus - The event contains the current status of one or more Transportation Sensor System objects.

ConfigChanged - One or more configuration values for the Transportation Sensor System have been changed.

ModeChanged - The communications mode of the TransportationSensorSystem has changed.

OpStatusChanged - The operational status of the TransportationSensorSystem has changed.

#### **5.2.1.1.31 TSSListEntry (Class)**

This struct is used to pass a TransportationSensorSystem object together with its ID. This struct is provided for convenience because when discovering an object, it is usually required to make a call to the object's getID() method.

#### **5.2.1.1.32 TSSStatus (Class)**

This class holds current status information for a TSS as follows:

m\_id - The ID of the TSS for which this status applies.

m\_zoneGrpTrafficParms - The traffic parameters for each ZoneGroup of the Transportation Sensor System as specified in the Sensor system's TSSConfiguration object.

m\_mode - The communication mode of the TSS.

m\_opStatus - The operational status for the TSS.

m\_trafficParameterTimestamp - A timestamp that records when the traffic parameter data was collected from the device.

m\_avgSpeed - average speed at the detector level.

m\_speedRange - speed range at the detector level (avg speed).

#### **5.2.1.1.33 UniquelyIdentifiable (Class)**

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

#### **5.2.1.1.34 ZoneGroup (Class)**

This class is used to group one or more detection zones of a Transportation Sensor System into a logical grouping. Traffic parameters for all detection zones included in the group are averaged to provide a single set of traffic parameters for the group.

#### **5.2.1.1.35 ZoneGroupTrafficParms (Class)**

This struct contains traffic parameters for a ZoneGroup.

m\_zoneGroupNumber - The number of the zone group for which the traffic parameters apply.

m\_trafficParms - The traffic parameter values for the zone group.

m\_zoneTrafficParms - zone parms for each zone in the group.

#### **5.2.1.1.36 ZoneTrafficParms (Class)**

This struct contains traffic parameters for a Zone.

m\_zoneNumber - The number of the zone for which the traffic parameters apply.

m\_trafficParms - The traffic parameter values for the zone.

## 5.3 GUI TSS Data Classes

### 5.3.1 Class Diagrams

#### 5.3.1.1 GUITSSDataClasses (Class Diagram)

This diagram shows objects related to adding TCP/IP connection functionality and geo location to TSS's.

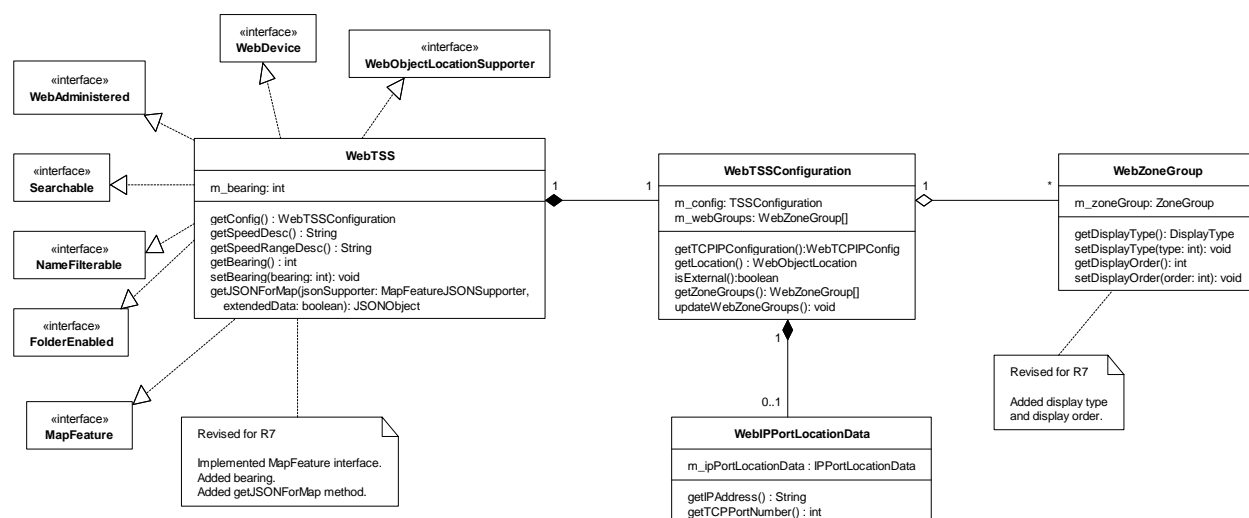


Figure 5-12 GUITSSDataClasses (Class Diagram)

##### 5.3.1.1.1 FolderEnabled (Class)

This interface provides access to information about an object that can be stored in a folder.

##### 5.3.1.1.2 MapFeature (Class)

This interface provides data necessary for displaying a feature on a map.

##### 5.3.1.1.3 NameFilterable (Class)

This java interface is implemented by classes which can be filter by name within the ObjectCache. A NameFilter object is passed into the ObjectCache to select NameFilterable objects in the cache.

##### 5.3.1.1.4 Searchable (Class)

This interface allows objects to be searched for via a substring search.

##### 5.3.1.1.5 WebAdministered (Class)

This interface allows the implementing class to be administered via the trader console

pages.

#### **5.3.1.1.6 WebDevice (Class)**

This interface contains common functionality for CHART devices.

#### **5.3.1.1.7 WebIPPortLocationData (Class)**

This class wraps the IPPortLocationData IDL structure and provides accessor methods to get the data. This class has data for identifying a TCP/IP address and port.

#### **5.3.1.1.8 WebObjectLocationSupporter (Class)**

This interface allows common processing for objects supporting an ObjectLocation via the WebObjectLocation wrapper class.

#### **5.3.1.1.9 WebTSS (Class)**

This class wraps the TransportationSystemSensor CORBA interface, caches data, and provides access to the cached data.

#### **5.3.1.1.10 WebTSSConfiguration (Class)**

This class wraps the TSSConfiguration IDL structure and provides accessors for easy access to the data.

#### **5.3.1.1.11 WebZoneGroup (Class)**

This class wraps the ZoneGroup IDL structure and provides accessors for easy access to the data.

## 5.4 Package chartlite.servlet.tss

### 5.4.1 Classs Diagrams

#### 5.4.1.1 chartlite.servlet.tss Classes

This diagram shows CHART GUI servlet classes related to traffic sensor signs.

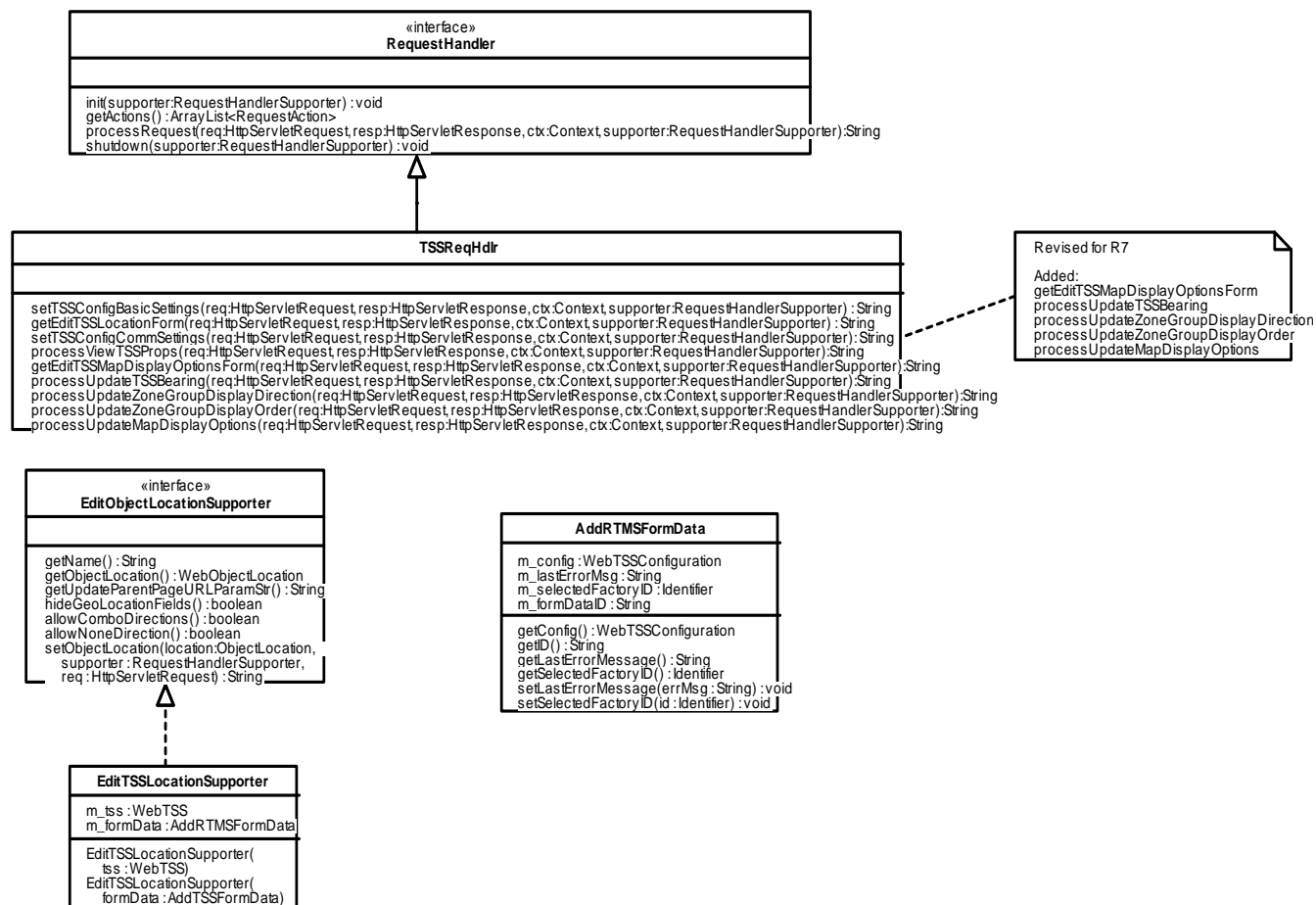


Figure 5-13 chartlite.servlet.tss\_classes (Class Diagram)

##### 5.4.1.1.1 AddRTMSFormData (Class)

This class represents the data in the Add RTMS form.

##### 5.4.1.1.2 EditObjectLocationSupporter (Class)

This interface provides functionality allowing the location data to be edited. (For example, the target of the edited location may be an existing object, or it may be a form data object for creating a new object).



#### **5.4.1.1.3 EditTSSLocationSupporter (Class)**

This class is used to support editing the location of an existing or new TSS.

#### **5.4.1.1.4 RequestHandler (Class)**

This interface specifies methods that are to be implemented by classes that are used to process requests.

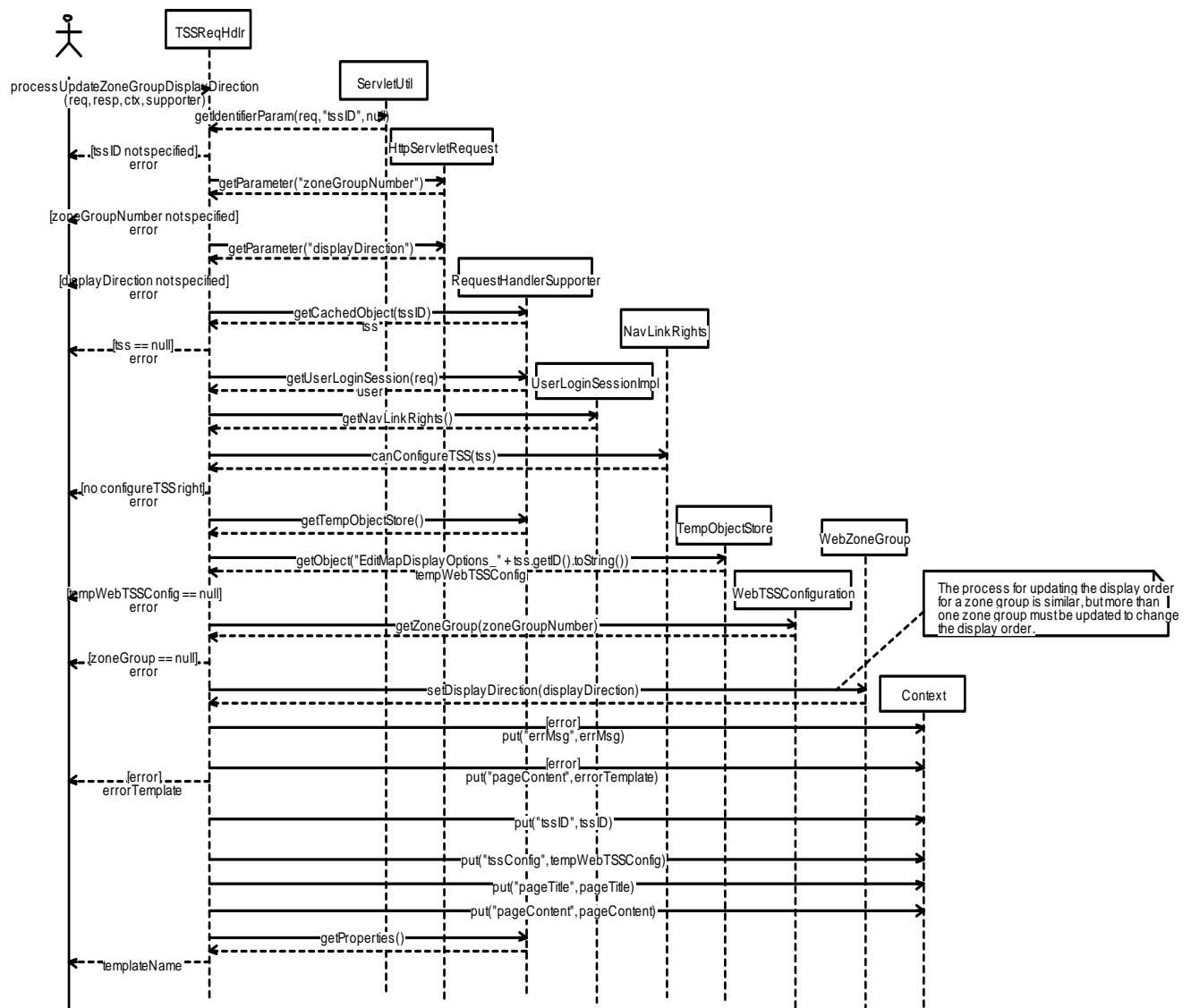
#### **5.4.1.1.5 TSSReqHdlr (Class)**

This class handles requests related to traffic sensor systems such as RTMS.

## 5.4.2 Sequence Diagrams

### 5.4.2.1 `chartlite.servlet.tss:processUpdateZoneGroupDisplayDirection`

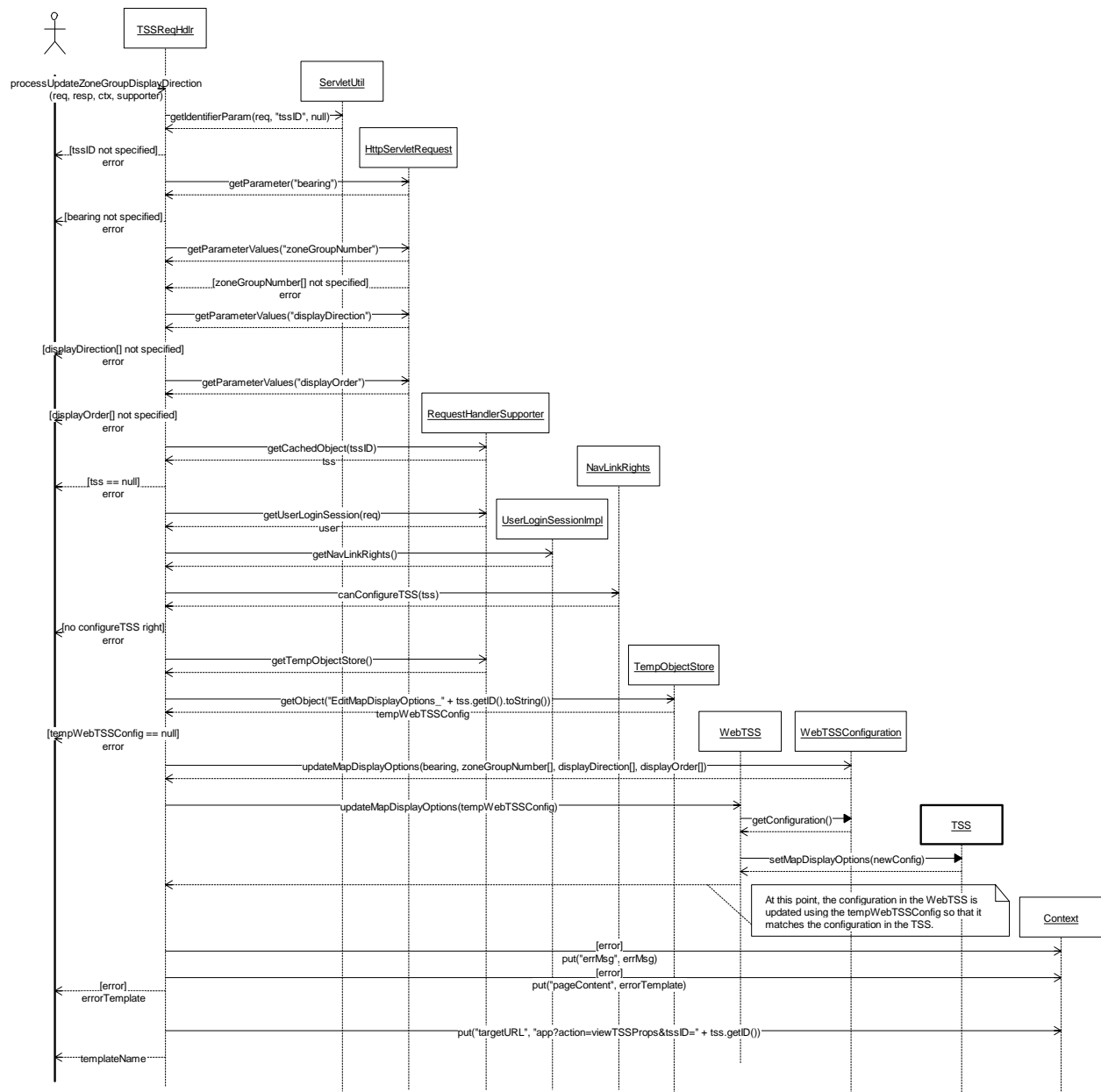
This diagram shows the process of updating the Display Direction for a TSS zone group in the servlet. The request parameters (`tssID`, `zoneGroupNumber`, and `displayDirection`) are parsed to get the ID of the TSS, the zone group number, and the new value of the display direction. If any of the parameters are missing, no further processing is done and an error is returned to the caller with an error template. Using the `tssID`, the TSS is retrieved from the cache. If the TSS is null, no further processing is done and an error is returned to the caller with an error template. The TSS is then used to check if the user has rights to configure a TSS. If the user does not have the rights to configure a TSS, no further processing is done and an error is returned to the caller with an error template. The TSS configuration is retrieved from the temporary object store using the ID of the TSS. If the TSS configuration is null, no further processing is done and an error is returned to the caller with an error template. The display direction is updated for the appropriate zone group (based on the zone group number) in the temporary TSS configuration. The Velocity context is populated with the required objects (the TSS ID, the temporary TSS configuration, the page title, and the page content) and the template name is returned to the caller.



**Figure 5-14** chartlite.servlet.tss:processUpdateZoneGroupDisplayDirection (Sequence Diagram)

#### 5.4.2.2 `chartlite.servlet.tss:processUpdateMapDisplayOptions`

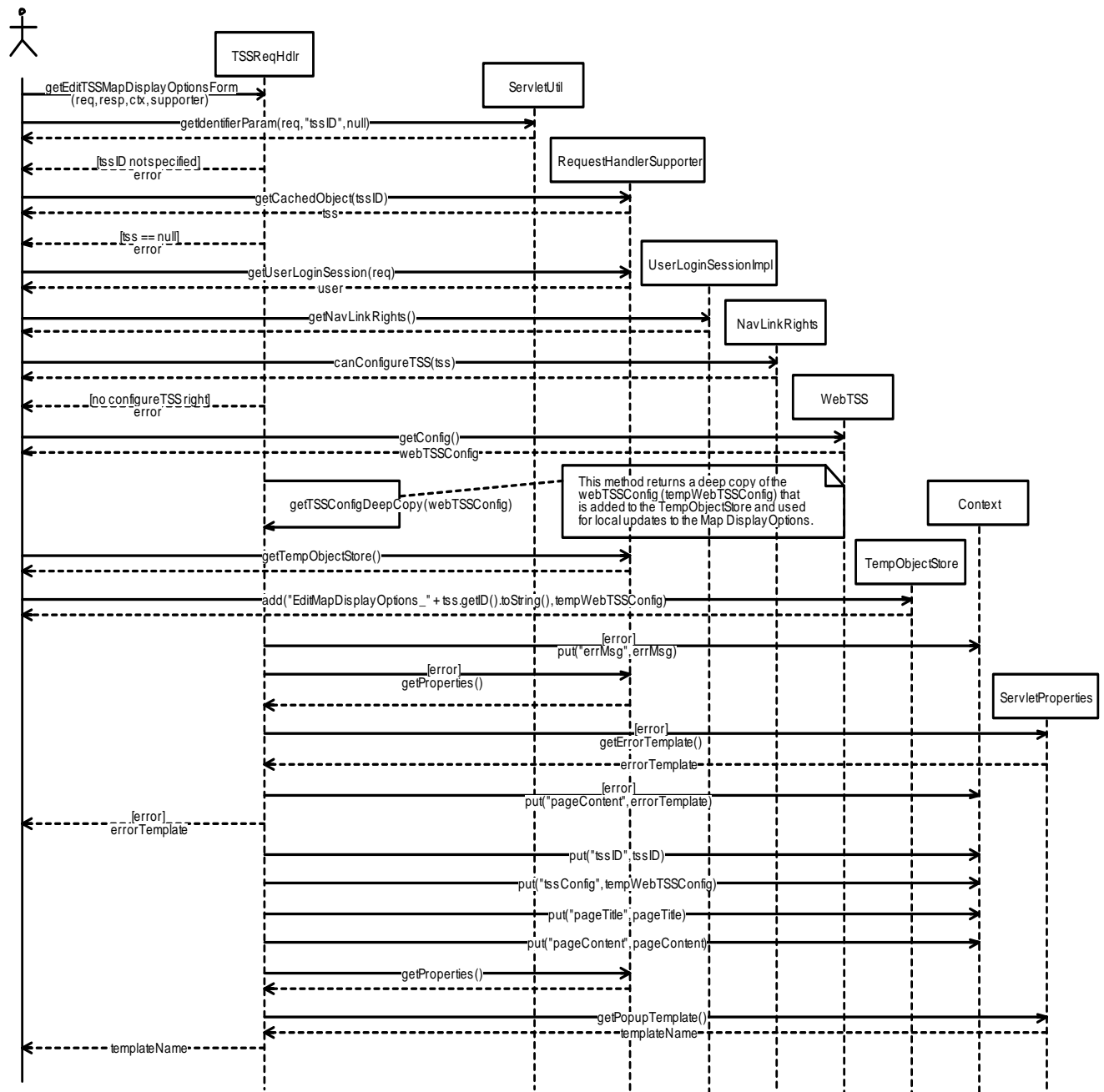
This diagram shows the process of updating the Map Display Options for a TSS in the servlet. The request parameters (`tssID`, `bearing`, `zoneGroupNumber`, `displayDirection`, and `displayOrder`) are parsed to get the ID of the TSS, the TSS bearing, the number for each zone group, the display direction for each zone group, and the display order for each zone group. If any of the parameters are missing, no further processing is done and an error is returned to the caller with an error template. Using the `tssID`, the TSS is retrieved from the cache. If the TSS is null, no further processing is done and an error is returned to the caller with an error template. The TSS is then used to check if the user has rights to configure a TSS. If the user does not have the rights to configure a TSS, no further processing is done and an error is returned to the caller with an error template. A user with the rights to configure a TSS can set the map display options in any mode (online, offline or maintenance mode). The TSS configuration is retrieved from the temporary object store using the ID of the TSS. If the TSS configuration is null, no further processing is done and an error is returned to the caller with an error template. The parameter values are used to update the temporary TSS configuration. The temporary TSS configuration is then passed to the Web TSS object to update its configuration. The WebTSS makes the CORBA call to the TSS object to set the map display options in the TSS Service. The Velocity context is populated with the required objects (the TSS ID, the temporary TSS configuration, the page title, and the page content) and the template name is returned to the caller.



**Figure 5-15 chartlite.servlet.tss:processUpdateMapDisplayOptions (Sequence Diagram)**

### 5.4.2.3 `chartlite.servlet.tss:getEditTSSMapDisplayOptionsForm`

This diagram shows the process of displaying the TSS Map Display Options Form in the servlet. The `tssID` request parameter is parsed to get the ID of the TSS. If this parameter is missing, no further processing is done and an error is returned to the caller with an error template. Using the `tssID`, the TSS is retrieved from the cache. If the TSS is null, no further processing is done and an error is returned to the caller with an error template. The TSS is then used to check if the user has rights to configure a TSS. If the user does not have the rights to configure a TSS, no further processing is done and an error is returned to the caller with an error template. A user with the rights to configure a TSS can set the map display options in any mode (online, offline or maintenance mode). The configuration is retrieved from the TSS and a deep copy of the configuration is obtained. The temporary configuration is added to the temporary object store using the ID of the TSS as a key. The Velocity context is populated with the required objects (the TSS ID, the temporary TSS configuration, the page title, and the page content) and the template name is returned to the caller.



**Figure 5-16 chartlite.servlet.tss:getEditTSSMapDisplayOptionsForm (Sequence Diagram)**

## 5.5 Package chartlite.servlet.map

### 5.5.1 Class Diagrams

#### 5.5.1.1 MapClasses (Class Diagram)

This diagram shows classes related to handling map-related requests.

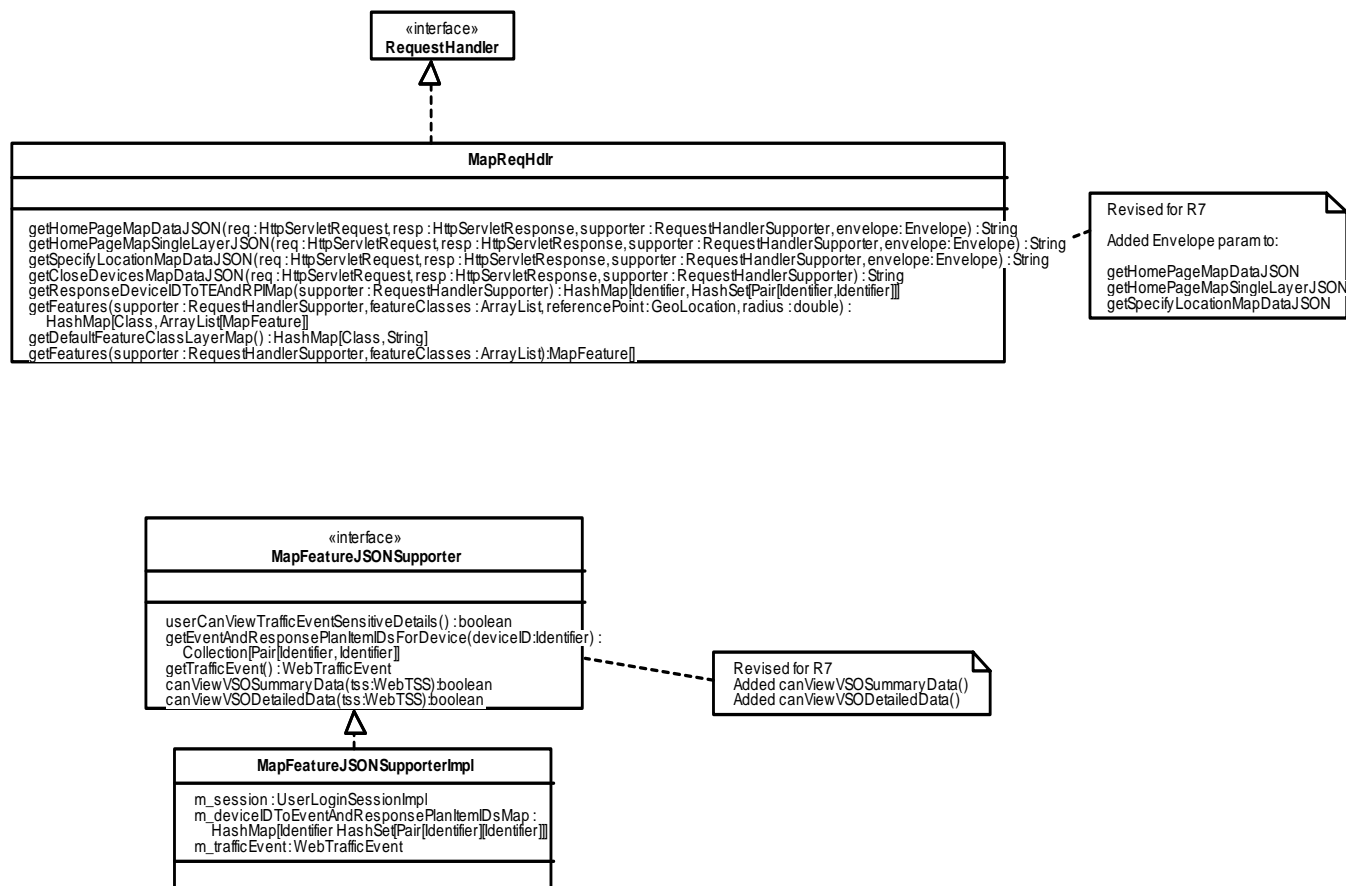


Figure 5-17 MapClasses (Class Diagram)

##### 5.5.1.1.1 MapFeatureJSONSupporter (Class)

This interface supplies information necessary for MapFeatures to build JSON for the map.

##### 5.5.1.1.2 MapFeatureJSONSupporterImpl (Class)

This class implements the MapFeatureJSONSupporter interface to supply data necessary for map features to build JSON necessary to represent their data.



#### **5.5.1.1.3 MapReqHdlr (Class)**

This class handles requests related to map functionality.

#### **5.5.1.1.4 RequestHandler (Class)**

This interface specifies methods that are to be implemented by classes that are used to process requests.



### 5.5.2.2 MapReqHdr:getTrafficEventJSON

This interface specifies methods that are to be implemented by classes that are used to process requests.

This diagram shows how the JSON for a Traffic Event is created. If the event is not open, not JSON will be created. If the centerEventsOnly flag is true, the controlling operations center for the Traffic Event and the user's operations center must match; otherwise, no JSON will be created. Once the JSON is obtained from the Traffic Event, the trafficEventsExtent Envelope object is expanded to include the location of the event. This Envelope is returned with the JSON to the map to describe the extent of the Traffic Events on the map. The JSON for the Traffic Event is then returned to the caller.

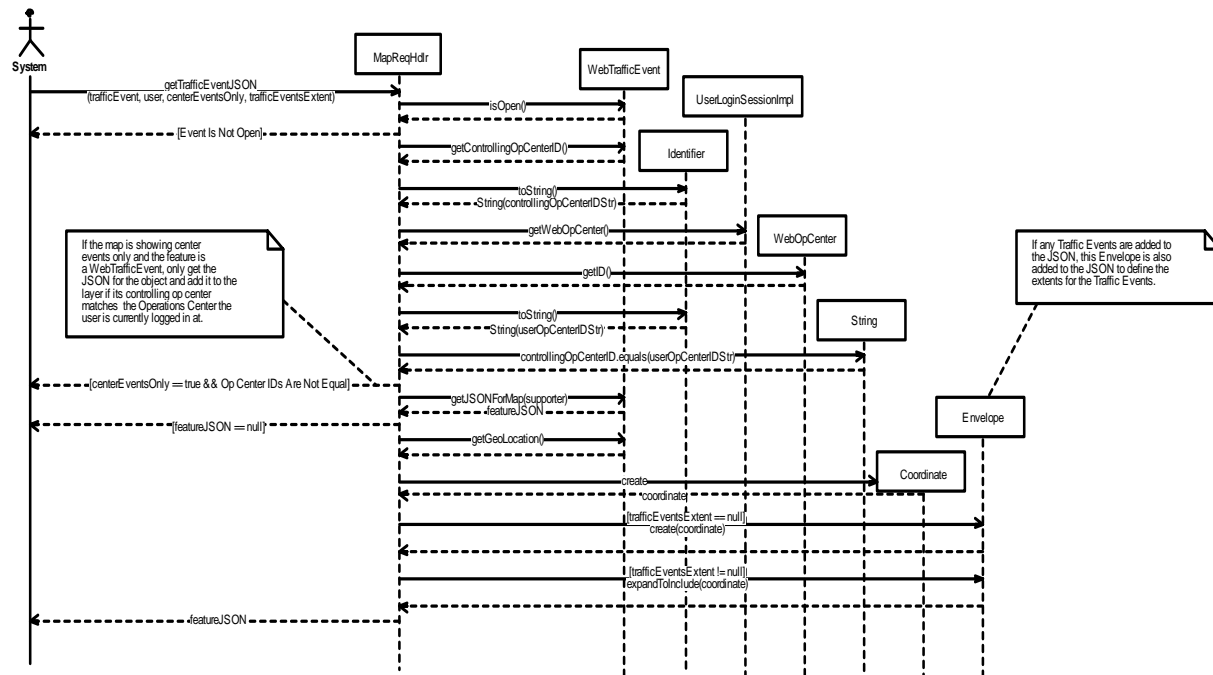
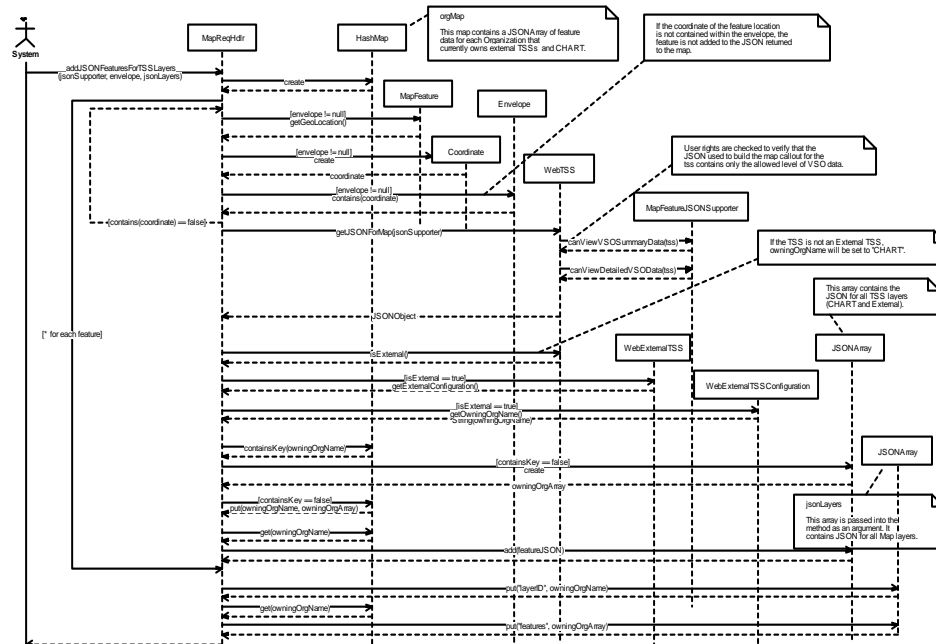


Figure 5-19 :getTrafficEventJSON (Sequence Diagram)

### 5.5.2.3 MapReqHdlr:addJSONFeaturesForTSSLayers

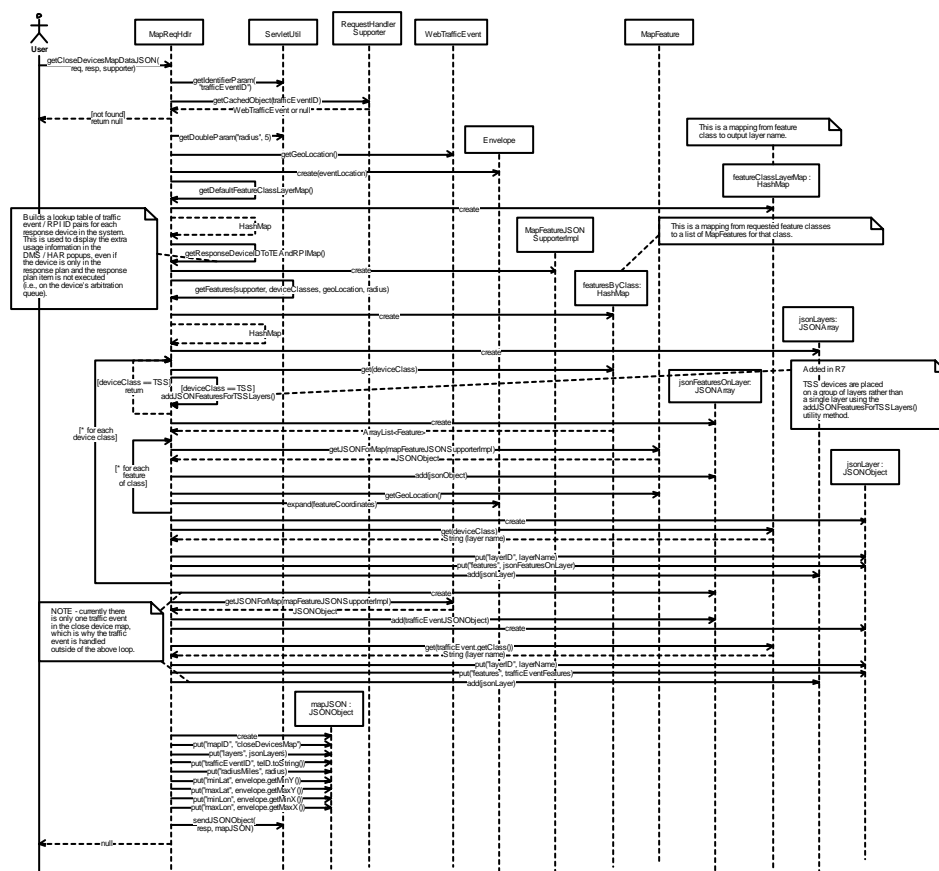
This diagram shows how the JSON for each TSS layer is added to the array containing JSON for all map layers. Since a TSS can be either a CHART TSS or an External TSS, a separate JSON array must be created for the CHART TSS layer and each of the External TSS layers. Each of these JSON arrays contains the JSON for all TSSs on that layer. The first step for each TSS is to determine if its location is contained in the envelope passed into the method as an argument. This envelope describes the current extent of the map and can be null if it is not applicable. If the envelope does not contain the TSS's location the JSON for the TSS is not included in the JSON array for the appropriate layer. When building the JSONObject for the TSS user rights are checked to ensure that the user is only provided the level of VSO data that they have been granted permission to see (if any). If the TSS is a CHART TSS, "CHART" is used for the owning organization name (and thus the layer name). If the TSS is an External TSS, the owning organization name is retrieved from the External TSS Configuration. The owning organization name is then used to store the JSON array for a specific layer in a HashMap for later retrieval. The JSON for each TSS in a specific layer is added to the JSON array for that layer in the HashMap. Once all TSSs have been processed, the JSON array for each TSS layer is added to the array containing JSON for all map layers. The array containing JSON for all map layers is then returned to the caller.



### Figure 5-20 :addJSONFeaturesForTSSLayers (Sequence Diagram)

#### 5.5.2.4 MapReqHdlr:getCloseDevicesMapDataJSON

This diagram shows how the request for the Close Devices Map JSON data is handled. The traffic event ID is used to retrieve the WebTrafficEvent, and the radius parameter is also parsed. An Envelope object is created to represent the extent of the traffic event and device locations. A HashMap is created to map the feature classes to layer names. Another HashMap is built for looking up the traffic events and response plan items for a given response device (DMS or HAR), which are displayed in the DMS/HAR callout. A call is made to get all MapFeature objects that are devices and are within the specified radius from the traffic event. Each class of devices is handled in a loop. If the device class is TSS, a utility method is called to create the layers based on whether the TSS is internal or external, and if it is external based on the owning Organization. If the device class is not TSS, the features of that class are retrieved from the HashMap and each feature is called to build a JSONObject, passing a context object that features may use to query necessary information. The JSONObject is then added to the list for the layer, and the envelope is expanded for the device's location. The traffic event is added on its own layer in the JSON data. Finally the layer data and other information are added to the JSON object representing all of the map data, and the JSON object is sent via the response.



### Figure 5-21 MapReqHdlr:getCloseDevicesMapDataJSON (Sequence Diagram)

## **5.6 Package CHART2.TSSManagementModule**

### **5.6.1 Class Diagrams**

#### **5.6.1.1 TSSManagementModulePkg**

This package manages all server activities related to Traffic Sensor Systems. Currently only Remote Traffic Microwave Sensor (RTMS) type devices are supported however it is designed to handle other TSS devices types. Devices are periodically polled (responding to a device-created event is not supported) and results are reported on CORBA event channels.



#### **5.6.1.1.1 AlertFactoryWrapper (Class)**

This singleton class provides a wrapper for the Alert Factory that provides automatic location of an Alert Factory and automatic re-discovery should the Alert Factory reference return an error. This class also allows for built-in fault tolerance by automatically failing over to a "working" Alert Factory without the user of this class being aware that this being done. In addition, this class defers the discovery of the Alert Factory until its first use, thus eliminating a start-up dependency for modules that use the Alert Factory.

This class delegates all of its method calls to the system AlertFactory using its currently known good reference to an AlertFactory. If the current reference returns a CORBA failure in the delegated call, this class automatically switches to another reference. When there are no good references (as is true the first time the object is used), this class issues a trader query to (re)discover the published Alert Factory objects in the system. During a method call, the trader will be queried at most one time and under normal circumstances, not at all.

#### **5.6.1.1.2 CommFailureDB (Class)**

This class is a utility used to log an entry in the Comm Failure log table in the database. This table is used to log details about any comm failure that occurs in the system.

#### **5.6.1.1.3 java.util.Timer (Class)**

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

#### **5.6.1.1.4 java.util.TimerTask (Class)**

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

#### **5.6.1.1.5 java.util.Vector (Class)**

A Vector is a growable array of objects.

#### **5.6.1.1.6 LogFile (Class)**

This class creates a flat file for writing system trace log messages and purges them at user specified interval. The log files created by this class are used for system debugging and maintenance only and are not to be confused with the system operations log which is modeled by the OperationsLog class.

#### **5.6.1.1.7 PolledTSSImpl (Class)**

This object implements the Transportation Sensor System interface as defined in IDL. This implementation provides the base functionality required for Transportation Sensor Systems that are polled periodically to retrieve traffic parameters. The only requirement for derived classes is to provide an implementation of the abstract poll method, which communicates



over a previously connected Port to obtain the traffic parameters from a TSS.

This implementation periodically polls the field device using the derived class implementation of the poll method. This implementation provides services such as raw data logging, averaging/summation of data into configured zone groups, asynchronous notification of configuration changes, and persistence/depersistence.

A DeviceFailure alert is created each time the device transitions into `HARDWARE_FAILURE`. Devices that cycle in and out of `HARDWARE_FAILURE` will send multiple DeviceFailure alerts so it is up to the AlertModule to prevent duplicate open DeviceFailure alerts for the same device.

#### **5.6.1.1.8 PortLocator (Class)**

The PortLocator is a utility class that helps one to connect to the port used by the device. The actual implementation of the operations is done by the derived classes depending on what protocol is used for communication.

#### **5.6.1.1.9 PushEventSupplier (Class)**

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

#### **5.6.1.1.10 RTMS (Class)**

The Remote Traffic Microwave Sensor (RTMS) is a detector manufactured by EIS, Inc. capable of providing lane level volume, speed, and occupancy data for up to 8 lanes of a roadway at a single location. This interface serves to identify TransportationSensorSystem objects as being of the type RTMS. It also provides a place holder for future operations that may not apply to TSS objects in general and are instead RTMS specific.

#### **5.6.1.1.11 RTMSDeviceStatus (Class)**

This class is used to pass raw data retrieved from the RTMS to the caller of the RTMSProtocolHdlr `getStatus()` method.

`m_trafficParameters` - the traffic parameters sensed by the device, such as volume, speed, and occupancy.

`m_healthStatus` - The health status byte reported from the RTMS. A value other than 10,

20, 30, 40, 50, 60, or 70 indicates a hardware problem.

m\_msgNum - The message number reported by the RTMS. This number is incremented sequentially when the RTMS dumps averaged data to a retrieval area at the end of a message period. It can be used to determine if the device is being polled too frequently or infrequently.

#### **5.6.1.1.12 RTMSFactory (Class)**

Objects which implement RTMSFactory are capable of adding an RTMS to the system.

#### **5.6.1.1.13 RTMSFactoryImpl (Class)**

This class implements the RTMSFactory interface as defined in the IDL. It holds all RTMSImpl objects that have been created within an instance of the RTMSManagementModule and allows for the addition and removal of RTMS objects. It also allows one to query all RTMS objects currently served from the factory.

This factory contains a timer that periodically fires, causing the RTMSFactoryImpl to collect the current status of each RTMSImpl and push the collective status in a single CORBA event.

#### **5.6.1.1.14 RTMSImpl (Class)**

This class is a derivation of the PolledTSSImpl that provides functionality for obtaining the current traffic parameters from an RTMS device. It makes use of an RTMSProtocolHandler to perform the device specific protocol to obtain the traffic parameters. It moves the data from the device specific format to the generic TSSPollResults object to allow the PolledTSSImpl to combine/average data based on zone group configuration, perform raw data logging, and other services that are common to Transportation Sensor System objects.

#### **5.6.1.1.15 RTMSProtocolHdlr (Class)**

This class is a utility that encapsulates the communication protocol of the RTMS device. It provides a high level method to get the status as an object. It formats a command and sends it to the device and receives and interprets the response from the device, passing the data back to the caller in the form of an RTMSDeviceStatus object.

#### **5.6.1.1.16 ServiceApplication (Class)**

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

#### **5.6.1.1.17 ServiceApplicationModule (Class)**

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

#### **5.6.1.1.18 TransportationSensorSystem (Class)**

A Transportation Sensor System (TSS) is a generic term used to describe a class of technology used for detection within the transportation industry. Examples of TSS devices range from the advanced devices, such as RTMS, to basic devices, such as single loop detectors.

This software interface is implemented by objects that provide access to the traffic parameters sensed by a Transportation Sensor System. Transportation Sensor Systems are capable of providing detection for one or more detection zones. A single loop detector would have one detection zone, while an RTMS would have 8 detection zones.

#### **5.6.1.1.19 TSSConfiguration (Class)**

This class holds configuration data for a transportation sensor system (TSS) as follows:

**m\_id** - The unique identifier for this TSS. This field is ignored when the object is passed to the TSS to change its configuration.

**m\_name** - The name used to identify the TSS.

**m\_location** - A descriptive location of the TSS.

**m\_dropAddress** - The drop address for the device.

**m\_bearing** - The bearing in degrees for displaying the TSS on the map. Valid values are from -1 to 359 (-1 = bearing not defined, 0 = East, 90 = North, 180 = West, and 270 = South). The default value is -1.

**m\_zoneGroups** - Logical groupings of detection zones, used to provide a single set of traffic parameters for one or more detection zones.

**m\_pollIntervalSecs** - The interval on which the TSS should be polled for its current traffic parameters (in seconds).

**m\_commPortCfg** - Communication configuration values.

**m\_portLocData** - Configuration information that determines which port manager(s) should be used to establish a connection with the SensorSystem.

**m\_debugComms** - Flag used to enable/disable the logging of communications data for this TSS. When enabled, command and response packets exchanged with the device are logged to a debugging log file.

#### **5.6.1.1.20 TSSConfigurationCopyResults (Class)**

This class is used to record successful configuration updates, errors, and warnings that happen while copying map display options from one TSSConfiguraiton object to another.

#### **5.6.1.1.21 TSSCurrentStatusPushTask (Class)**

This class is a timer task that is executed on a regular interval. When this task is run, it calls into the RTMSFactoryImpl object to have it collect the status for all RTMSImpl objects and to push a CurrentStatus event with the collected data.

#### **5.6.1.1.22 TSSDBData (Class)**

This class holds data that is retrieved from the database during start-up for a Transportation Sensor System object that existed in the system during a prior run of the software.

#### **5.6.1.1.23 TSSEvent (Class)**

This class is a CORBA union that contains varying data depending on the current value of the discriminator.

If the discriminator is ConfigChanged, this union contains a TSSConfig object.

If the discriminator is ObjectAdded, this union contains an ObjectAddedEventInfo object.

If the discriminator is ObjectRemoved, this union contains a byte[] containing the unique identifier for the Traffic Sensor System that was removed.

If the discriminator is CurrentStatus the union contains an array of one or more TSSStatus objects.

If the discriminator is ModeChanged, the union contains a ModeChangedEventInfo.

If the discriminator is OpStatusChanged, the union contains an OpStatusChangedEventInfo object.

#### **5.6.1.1.24 TSSManagementDB (Class)**

This class is a utility that provides methods for adding, removing, and updating database data pertaining to Transportation Sensor Systems. Because this class is designed to be generic and work for RTMS as well as other TSS derived objects, the add method requires a model id to be passed. This allows data for a specific model to be retrieved by model specific factories during system initialization.

#### **5.6.1.1.25 TSSManagementModulePkg (Class)**

This class is a ServiceApplicationModule used to serve an RTMSFactory object. The RTMSFactory serves zero or more RTMS objects. By providing an implementation of the ServiceApplicationModule interface, this class can be included in the CHART2 service

application framework, which provides common services needed to serve CORBA objects within the CHART 2 system.

#### **5.6.1.1.26 TSSManagementProperties (Class)**

This class provides a wrapper to the application's properties file that provides easy access to the properties specific to the TSSManagementModule. These properties include the name of the file where raw traffic parameter data is to be logged, the directory where debug log files are to be kept, and the interval at which the status of all TSS objects is to be collected and pushed in a CORBA event.

#### **5.6.1.1.27 TSSPollingTask (Class)**

This class is a TimerTask that is used by an RTMS to schedule its asynchronous polling with a Timer object.

#### **5.6.1.1.28 TSSPollResults (Class)**

This class is a data holder used to pass the results of device polling from the PolledTSSImpl derived class back to the base class for processing. The traffic parameter data passed is lane (detection zone) level. The operational status is the status as determined by the derived class.

m\_trafficParms - An array of traffic parameters for the current poll cycle, with one array entry for each detection zone of the device.

m\_opStatus - The operational status as determined by the derived class.

#### **5.6.1.1.29 TSSStatus (Class)**

This class holds current status information for a TSS as follows:

m\_id - The ID of the TSS for which this status applies.

m\_zoneGrpTrafficParms - The traffic parameters for each ZoneGroup of the Transportation Sensor System as specified in the Sensor system's TSSConfiguration object.

m\_mode - The communication mode of the TSS.

m\_opStatus - The operational status for the TSS.

m\_trafficParameterTimestamp - A timestamp that records when the traffic parameter data was collected from the device.

m\_avgSpeed - average speed at the detector level.

m\_speedRange - speed range at the detector level (avg speed).

## 5.6.2 Sequence Diagrams

### 5.6.2.1 PolledTSSImpl:setMapDisplayOptions

This diagram shows the processing performed when a client process calls the TSS.setMapDisplayOptions method. First a check is made to verify that the caller has the configureTSS functional right. If not, an AccessDenied exception is thrown. If the user does have rights, the copyMapDisplayOptionsToCurrentConfig() helper method is called passing the new TSSConfiguration that contains the desired map display options. This method will take a synchronization lock on the current configuration, and then will copy the new TSS bearing to the configuration. It will then loop over all zone groups in the new configuration and set the display type and display order for each. For each value that is updated in the current TSSConfiguration the method will log a success to a MapDisplayOptionsCopyResults object. For each member that cannot be set it will record a warning or error as appropriate. If an error occurs, this method will leave the original configuration unchanged. The MapDisplayOptionsCopyResults object is then returned to the caller. The caller checks for errors, if there are any a CHART2Exception is thrown with the error information. If there are none, a list of successfully copied members is obtained from the copy results object. If at least one member was copied successfully, the database class updateConfig() method will be called to persist the new configuration values, a list of changed elements will be logged to the operations log and a ConfigChanged CORBA event will be pushed. Regardless of successes, a check is next made to see if there were any warnings. If so, they are returned to the caller.

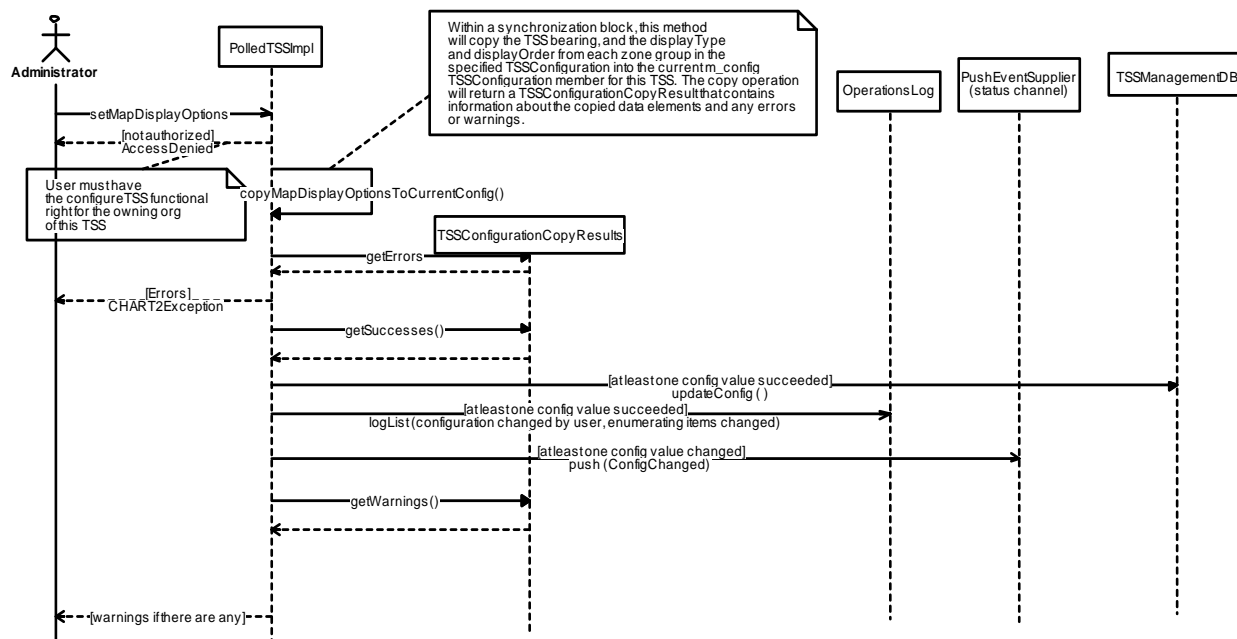
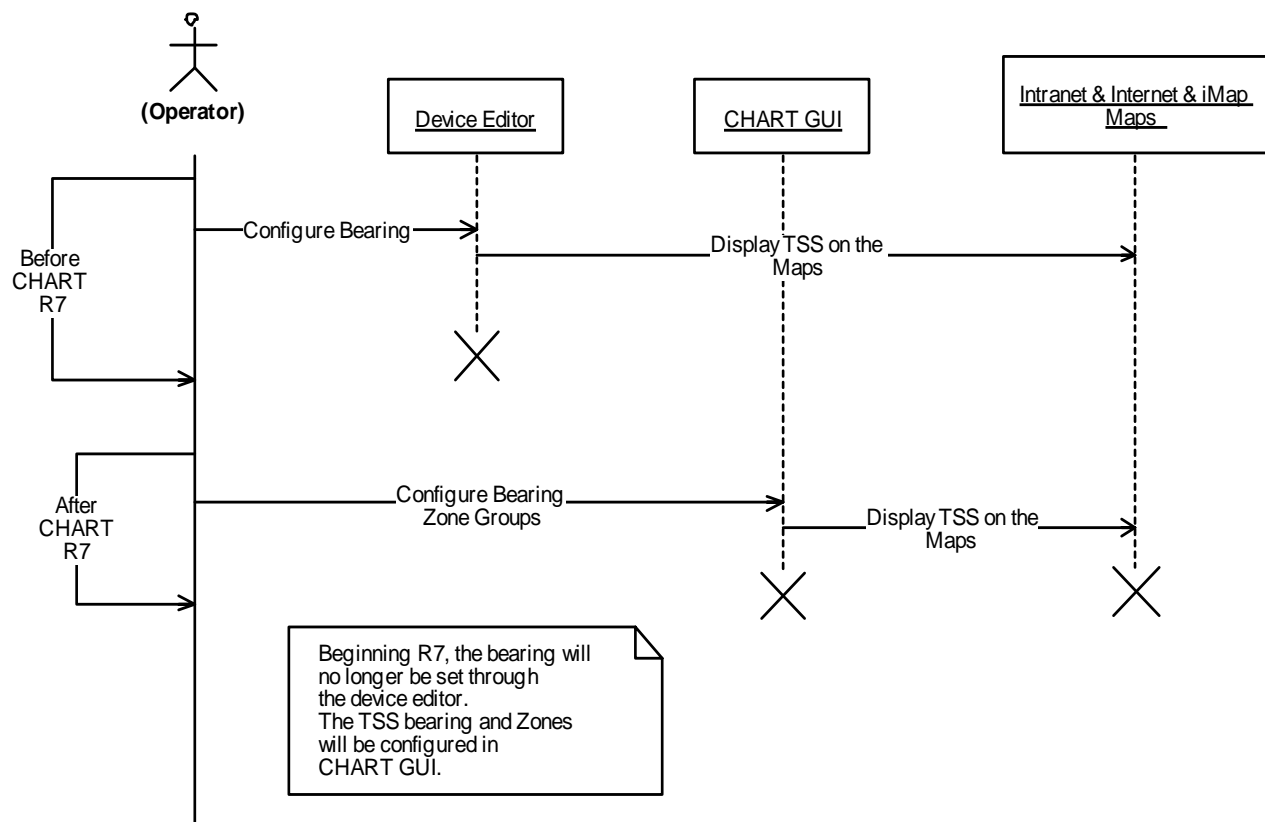


Figure 5-23 PolledTSSImpl:setMapDisplayOptions (Sequence Diagram)

## 5.7 Mapping Device Editor

### 5.7.1 Sequence Diagrams

#### Device Editor



**Figure 5-24 DataSynchronization: HumanMachine**

Beginning CHART R7 (Mapping R6), the Integrated Map in the CHART application will handling the mapping of CHART Devices (DMS, HAR, SHAZAM, CAMERA, DETECTOR); Device viewing, adding, updating and removing will no longer be available in the CHART Mapping Device Editor.

## 5.8 CHART Intranet & Internet Mapping GUI

There are no changes have been made in CHART Intranet and Internet Mapping GUI to accommodate this feature. The change for this feature is described in the device editor.

## 5.9 CHART Data Exporter Synchronization (CHART Intranet Map)

### 5.9.1 Class Diagrams

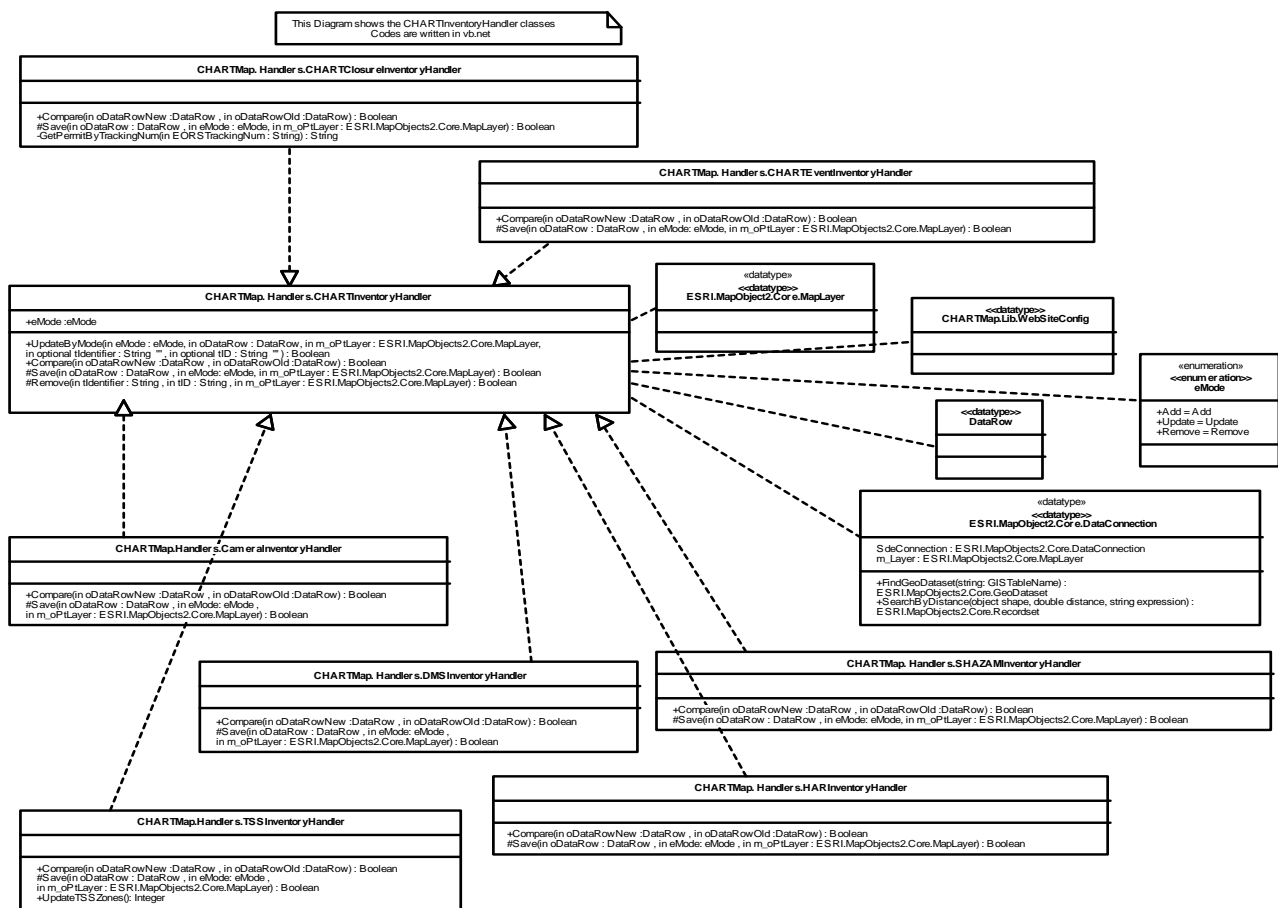


Figure 5-25 Data Exporter Synchronization (Class Diagram)



### **5.9.1.1 CHARTInventoryHandler Classes**

#### **5.9.1.2 CHARTMap.Handlers.CHARTInventoryHandler (Class)**

This is a base class for inventory updates. This class contains methods to determine the update status (add, update, or remove). In addition, this class contains a method to remove an object from the spatial table. Derived classes must implement Compare() and Save().

#### **5.9.1.3 CHARTMap.Handlers.DMSInventoryHandler (Class)**

This class extends the CHARTInventoryHandler class and implements the Compare() and Save() methods. This class is used to compare DMS records between the spatial and non-spatial table. This class is also used to update or add new record(s) to the DMS spatial table.

#### **5.9.1.4 CHARTMap.Handlers.HARInventoryHandler (Class)**

This class extends the CHARTInventoryHandler class and implements the Compare() and Save() methods. This class is used to compare HAR records between the spatial and non-spatial table. This class is also used to update or add new record(s) to the HAR spatial table.

#### **5.9.1.5 CHARTMap.Handlers.SHAZAMInventoryHandler (Class)**

This class extends the CHARTInventoryHandler class and implements the Compare() and Save() methods. This class is used to compare SHAZAM records between the spatial and non-spatial table. This class is also used to update or add new record(s) to the SHAZAM spatial table.

#### **5.9.1.6 CHARTMap.Handlers.CHARTEventInventoryHandler (Class)**

This class extends the CHARTInventoryHandler class and implements the Compare() and Save() methods. This class is used to compare CHART Event records between the spatial and non-spatial table. This class is also used to update or add new record(s) to the CHART Event spatial table.

#### **5.9.1.7 CHARTMap.Handlers.CHARTClosureInventoryHandler (Class)**

This class extends the CHARTInventoryHandler class and implements the Compare() and Save() methods. This class is used to compare CHART Closure records between the spatial and non-spatial table. This class is also used to update or add new record(s) to the CHART Closure spatial table.

#### **5.9.1.8 CHARTMap.Handlers.CameraInventoryHandler (Class)**

This class extends the CHARTInventoryHandler class and implements the Compare() and Save() methods. This class is used to compare Camera records between the spatial and non-spatial table. This class is also used to update or add new record(s) to the Camera spatial table.

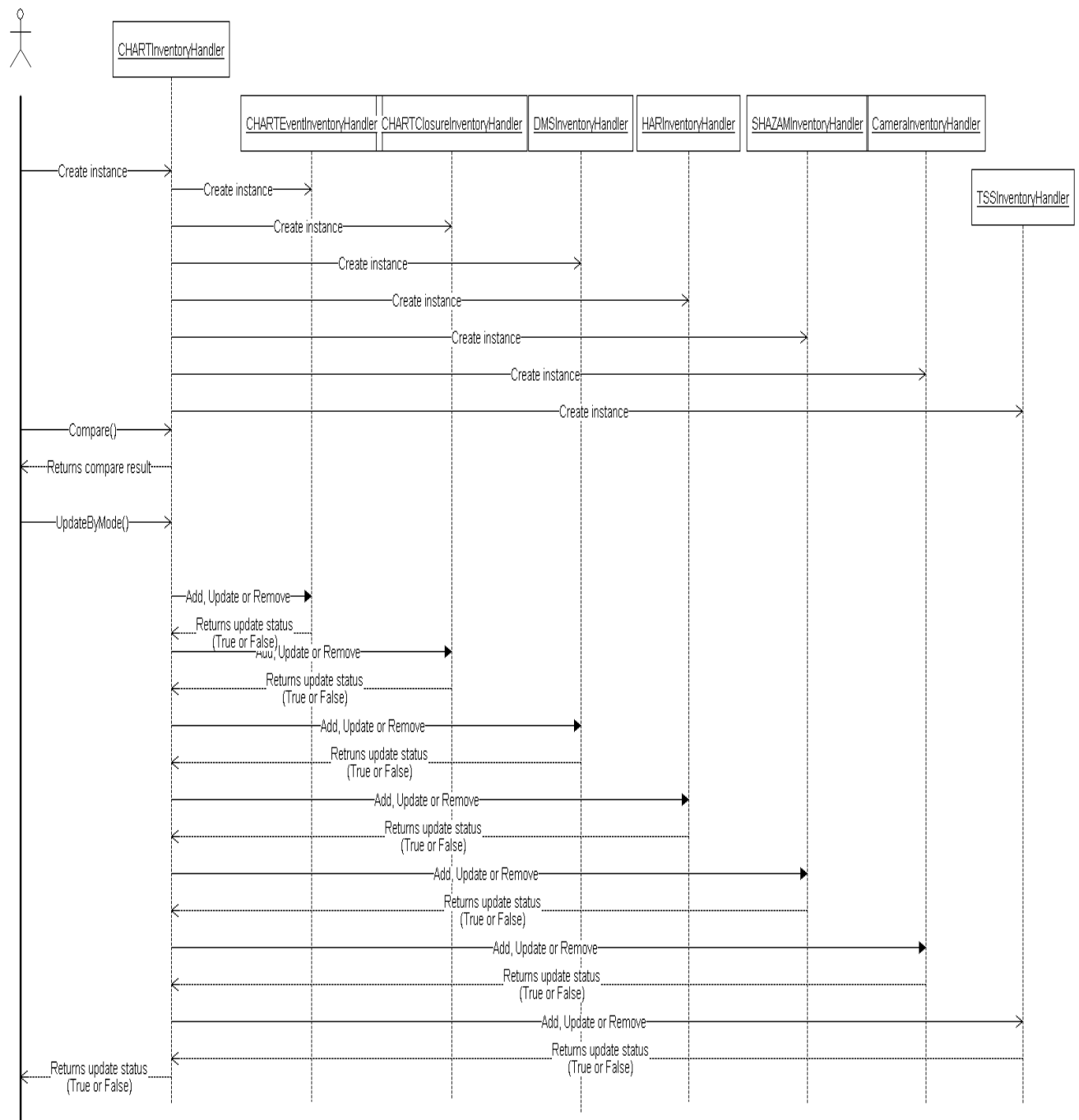
### 5.9.1.9 CHARTMap.Handlers.TssInventoryHandler (Class)

This class extends the CHARTInventoryHandler class and implements the Compare() and Save() methods. This class is used to compare detector records between the spatial and non-spatial table. This class is also used to update or add new record(s) to the detector spatial table.

## 5.9.2 Sequence Diagram

### 5.9.2.1 CHART Data Exporter Synchronization

This diagram shows the processing that occurs when a request to synchronize CHART Events or Devices. The process starts by parsing the incoming query string and determines which object to be synchronized. The process also writes each process events into the log file in the local machine. An associated object is created once the process determines which object to be synchronized. Then it calls the UpdateInventory method to start comparing each column of the spatial and the non-spatial tables based on the distinct identifier. The existing spatial record is updated by calling the UpdateByMode method if the process finds a difference between the two tables. As an exception, a record will be removed by calling the UpdateByMode method if the non-spatial table contains 0 or Null value for both Latitude and Longitude columns which indicates the Events or Device has been un-mapped from the integrated map. If a record existed in the non-spatial table but it does not exist in the spatial table, the process will then add a new record by calling the UpdateByMode method to the spatial table unless the value of the Latitude and Longitude columns are either 0 or Null. The process will remove the record from the spatial table if the record by calling the UpdateByMode method only existed in the spatial table.



**Figure 5-26 Data Exporter Synchronization (Sequence Diagram)**

## 6 Use Cases – NTCIP Camera

The use case diagrams depict new functionality for the CHART NTCIP Camera functionality and also identify existing features that will be enhanced. The use case diagrams for this feature exist in the Tau design tool in the Release7 area. The sections below indicate the title of the use case diagrams that apply to this feature.

### 6.1 R7 Camera Use Cases

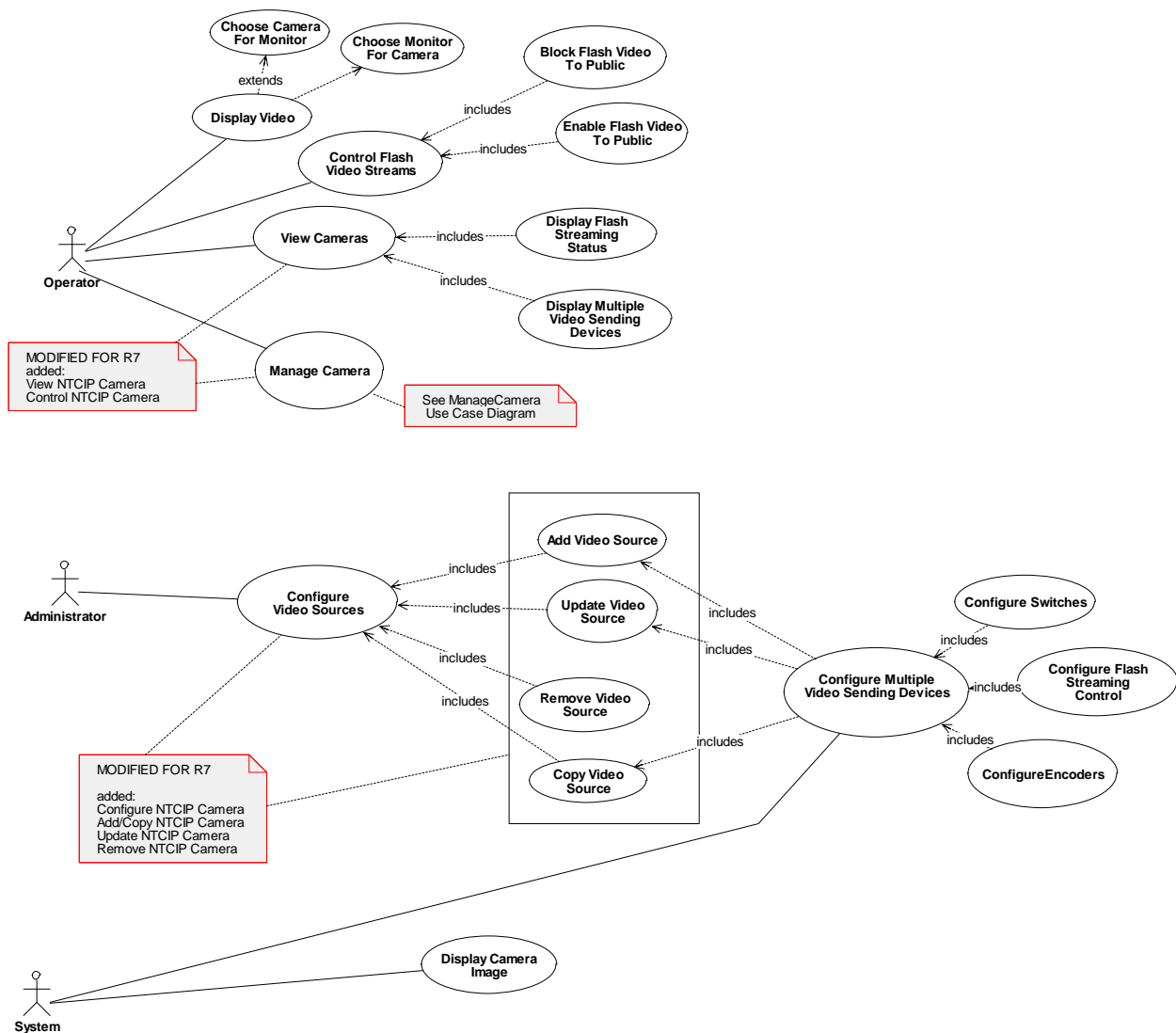


Figure 6-1. R7CameraUses (Use Case Diagram)

### **6.1.1 Add Video Source (Use Case)**

The system shall allow the user to add video sources to the system. Video Sources include generic unspecified video sources, "No Video Available" sources, fixed cameras, and controllable cameras including COHU, Vicon, and NTCIP cameras. The system allows an administrator to configure multiple video sending devices for the video source.

### **6.1.2 Administrator (Actor)**

An administrator is a CHART user that has functional rights assigned to allow them to perform administrative tasks, such as system configuration and maintenance.

### **6.1.3 Block Flash Video To Public (Use Case)**

An operator shall be able to block a camera's flash stream to the public.

### **6.1.4 Choose Camera For Monitor (Use Case)**

An operator shall be able to choose a camera to display on a monitor from the monitor list. The monitors shown in the list should have the correct route displayed. If a camera contains a local and routed connecting device, local will be shown.

### **6.1.5 Choose Monitor For Camera (Use Case)**

An operator shall be able to choose a monitor for display from the camera list. The monitors shown in the list should have the correct route displayed. If the monitor contains a local and routed connecting device, local will be shown.

### **6.1.6 Configure Video Sources (Use Case)**

The system allows an administrator with the Configure Camera right to configure video sources in the CHART system. Video Sources include generic unspecified video sources, "No Video Available" sources, fixed cameras, and controllable cameras including COHU, Vicon, and NTCIP cameras. The system allows an administrator to configure multiple video sending devices for the video source.

### **6.1.7 Configure Flash Streaming Control (Use Case)**

The system shall allow an administrator to specify a flash video stream control for the video source. (This is the system which manages the "red button", also known as the flash "kill switch".)

### **6.1.8 Configure Multiple Video Sending Devices (Use Case)**

An administrator shall be able to configure one or more video sending devices and flash video stream controls for each video source in the system..

#### **6.1.9 Configure Switches (Use Case)**

The system shall allow an administrator to configure one or more switches as a video sending device for a camera.

#### **6.1.10 Configure Encoders (Use Case)**

The system shall allow an administrator to configure one or more encoders as a video sending device for a camera.

#### **6.1.11 Control Flash Video Streams (Use Case)**

An operator shall be able to control a camera's flash stream to the public.

#### **6.1.12 Copy Video Source (Use Case)**

The system shall allow a user to copy video sources when creating new video sources. Video Sources include generic unspecified video sources, "No Video Available" sources, fixed cameras, and controllable cameras including COHU, Vicon, and NTCIP cameras. The system shall allow an administrator to copy multiple (one or more) video sending devices while copying a video source..

#### **6.1.13 Display Camera Image (Use Case)**

When the system displays a camera image on a monitor, the correct sending device will be used based on the receiving device's video fabric.

#### **6.1.14 Display Flash Streaming Status (Use Case)**

An operator shall be able to view the streaming status of a camera.

#### **6.1.15 Display Multiple Video Sending Devices (Use Case)**

The operator shall be able to view multiple sending device configurations for a video camera.

#### **6.1.16 Display Video (Use Case)**

An operator shall be able to display video when a camera has more than one sending device specified.

#### **6.1.17 Enable Flash Video to Public (Use Case)**

An operator shall be able to enable a camera's flash stream to the public.

#### **6.1.18 Manage Camera (Use Case)**

An operator with the correct functional rights may perform basic operations on a camera. Please refer to the Manage Camera Use Case diagram for more detailed information.

#### **6.1.19 Operator (Actor)**

An operator is a user of the system who has been assigned a valid username/password combination and granted roles for system access.

#### **6.1.20 Remove Video Source (Use Case)**

The system shall allow a user to remove video sources from the system.

#### **6.1.21 System (Actor)**

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use case that another actor has initiated.

#### **6.1.22 Update Video Source (Use Case)**

The system shall allow a user to update video source attributes. Video Sources include generic unspecified video sources, "No Video Available" sources, fixed cameras, and controllable cameras including COHU, Vicon, and NTCIP cameras. The system shall allow an administrator to configure multiple (one or more) video sending devices while updating a video source.

#### **6.1.23 View Cameras (Use Case)**

An operator shall be able to view the details for a camera. Details include the flash streaming configuration/status and multiple video sending devices.

## 6.2 DisplayCamera (Use Case Diagram)

An operator may display any camera on any monitor subject to certain restrictions. First the operator must have the proper functional rights to display a camera on a monitor. Next, the operator must have the proper functional rights to display a particular camera. Finally, that camera must be online. An operator may display a local camera on a local monitor, a remote camera on a local monitor, a local camera on a remote monitor, or a remote camera on a remote monitor. A local camera is a camera homed to the same server as the operator's workstation. A local monitor refers to a monitor in the requesting operator's monitor group. An operator with the correct functional rights may also start and stop a camera tour running on a local or remote monitor. A display request may also entail displaying a camera on a monitor across switch fabrics and using a router to manage the limited number of connections between the switch fabrics.

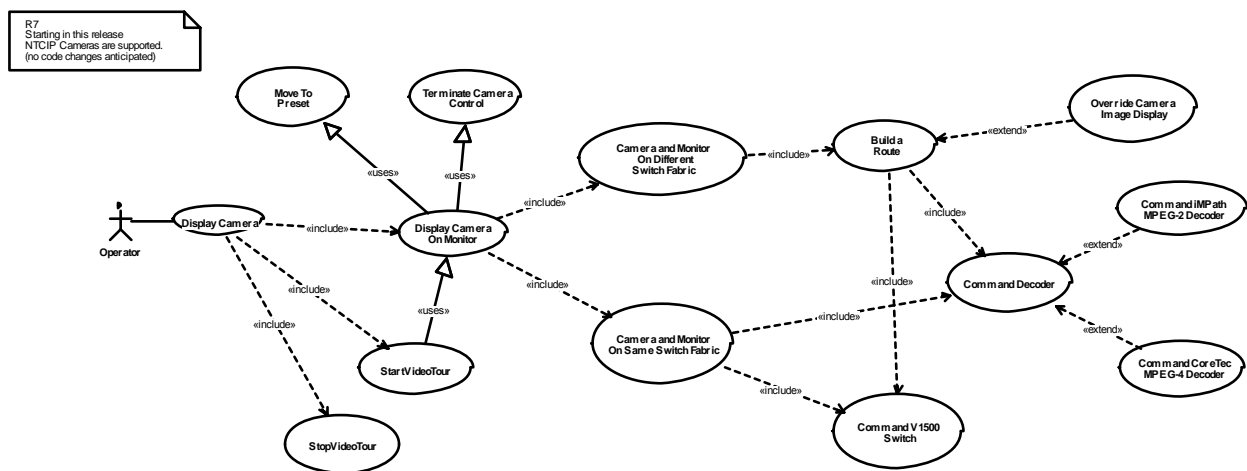


Figure 6-2 Display Camera (Use Case Diagram)

### 6.2.1 Build a Route (Use Case)

The Router tracks all routes between switch fabrics. The limited number of connections is managed. This means that all requests are evaluated based on a set of business rules, and fulfilled if possible. It is possible, based on the business rules, that a current image, using one of the target routes, will need to be overridden. The Router will command either the V1500 Switch or the Decoder as it builds the legs of the route.

### 6.2.2 Camera and Monitor On Different Switch Fabric (Use Case)

After a display request has been evaluated and it has been determined that the camera and monitor are on different switch fabrics, a route between the camera and monitor must be



computed and evaluated.

### **6.2.3 Camera and Monitor On Same Switch Fabric (Use Case)**

After a display request has been evaluated and it has been determined that both the camera and monitor are on the same switch fabric, the receiving device can be commanded directly. For IP based cameras and monitors, a Decoder is commanded. For V1500 based cameras and monitors, a V1500 switch is commanded.

### **6.2.4 Command CoreTec MPEG-4 Decoder (Use Case)**

In order to accomplish the task on displaying a camera on a monitor attached to a CoreTec MPEG-4 decoder, the system will command an IP based CoreTec MPEG-4 decoder to perform the video switching. The decoder will actually stop receiving the video stream for the current camera and start receiving the video stream for the new camera. It will do so by dropping the multicast group associated with the current camera's video stream and joining the multicast group associated with the new camera's video stream.

### **6.2.5 Command Decoder (Use Case)**

In order to accomplish the task on displaying a camera on a monitor, the system will command an IP based decoder to perform the video switching. The decoder will actually stop receiving the video stream for the current camera and start receiving the video stream for the new camera. It will do so by dropping the multicast group associated with the current camera's video stream and joining the multicast group associated with the new camera's video stream.

### **6.2.6 Command iMPath MPEG-2 Decoder (Use Case)**

In order to accomplish the task on displaying a camera on a monitor attached to an iMPath MPEG-2 decoder, the system will command an IP based iMPath MPEG-2 decoder to perform the video switching. The decoder will actually stop receiving the video stream for the current camera and start receiving the video stream for the new camera. It will do so by dropping the multicast group associated with the current camera's video stream and joining the multicast group associated with the new camera's video stream.

### **6.2.7 Command V1500 Switch (Use Case)**

A V1500 Switch is commanded whenever a source and a sink on a V1500 switch need to be connected.

### **6.2.8 Display Camera (Use Case)**

An operator with the correct functional rights may display a camera on a monitor. See the Display Camera use case diagram for a more detailed explanation.

### **6.2.9 Display Camera On Monitor (Use Case)**

An operator with the proper functional rights may display a camera on a monitor by commanding the proper Decoder or V1500 Switch. If the camera currently displayed on the target monitor is being controlled, and that monitor the only display within the controlling operator's monitor group, the display request will be normally rejected. The exception to this rule occurs if a camera is being taken offline, and the camera is being controlled. In this case a NoVideoAvailable source is displayed on the monitor and camera control is terminated.

### **6.2.10 Move To Preset (Use Case)**

When the last image of a camera is removed from any monitor (as part of a new display request), the camera will move to a default preset position if defined. A camera may also move to a pre-defined preset as part of a display associated with a video tour.

### **6.2.11 Override Camera Image Display (Use Case)**

The Override Camera Image use case deals with the situation where a high priority display request comes into the system when there is no currently available route to fulfill the request. In such a case one of routes currently in use will have to be taken for use by the higher priority request. This means that a “No Video Available” source will be displayed on the monitor(s) that have had their camera image overridden.

### **6.2.12 StartVideoTour (Use Case)**

An operator with the proper functional rights may start a video tour on the selected monitor. The video tour list is defined in the CHART II database. The video tour list consists of a list of cameras to be displayed in succession for a configurable dwell time.

### **6.2.13 StopVideoTour (Use Case)**

An operator with the proper functional rights may stop a video tour running on the selected monitor. The operator need not be the operator who started the camera tour.

### **6.2.14 Terminate Camera Control (Use Case)**

An operator with the proper functional rights may manually terminate a camera control session that the operator is actively using. Note that an operator who has the proper functional rights to establish the control session will always have the proper functional rights to terminate that session. Also, a camera control session may be terminated if that session is overridden by an appropriately privileged operator. Also, an active control session may be terminated if a camera is taken offline or if the camera is no longer displayed on a monitor within the controlling operator's monitor group as a result of displaying a NoVideoAvailable source. Note that part of this process will include terminating the camera control GUI, although that is beyond the scope of this document.

## 6.3 MaintainCamera (Use Case Diagram)

This diagram shows use cases related to maintaining Cameras via the Maintenance GUI.

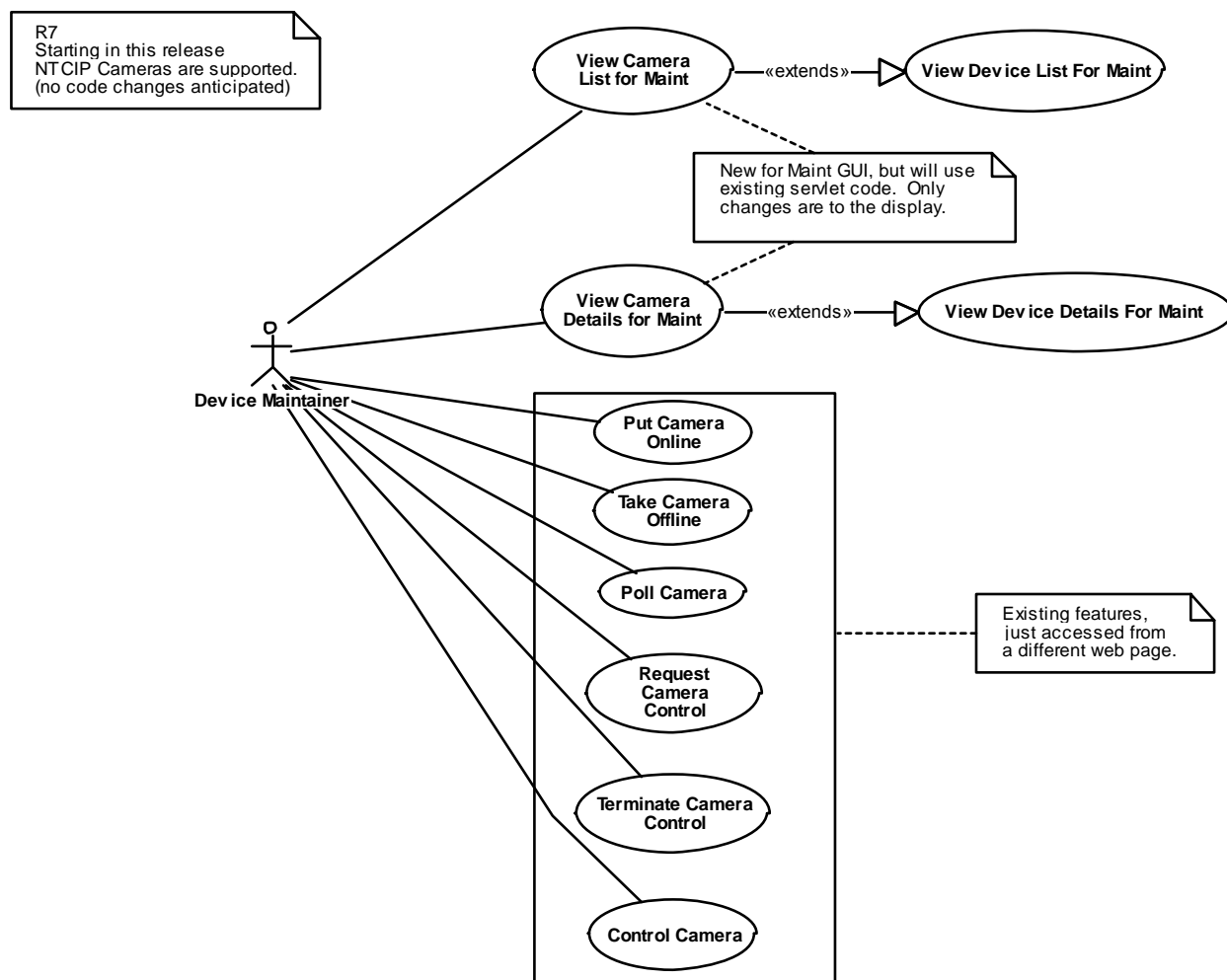


Figure 6-3 MaintainCamera (Use Case Diagram)

### 6.3.1 Control Camera (Use Case)

An operator with the proper functional rights may control a camera. In this case, control refers to issuing commands to the camera to cause the camera to pan/tilt/zoom etc. Refer to the send Camera Commands use case diagram for more details.

### 6.3.2 Poll Camera (Use Case)

A camera is polled by the system in order to establish the status of the camera control communications path. The polling process consists of sending the camera a poll command

and receiving a response from the camera. This is done by the system for all cameras that are online, regardless of whether the cameras are controlled or not. It is also done immediately after camera control has been granted so that the camera control status is current. In addition, polling takes place while a camera is actively controlled. When a camera is actively controlled, the polling is typically much more frequent than when the camera is not actively controlled.

### **6.3.3 Put Camera Online (Use Case)**

An operator with the proper functional rights can put a camera online if the camera is currently offline. Putting the camera online makes it available for display and control to any operators having the proper functional rights.

### **6.3.4 Request Camera Control (Use Case)**

An operator with the proper functional rights may request control of a camera. This means that the operator may send pan/tilt/zoom (PTZ) and other commands to the camera. The system evaluates the request, and will accept the request, prompt the operator to override an existing camera control session, or reject the request. If the request is accepted or the user chooses to override an existing control session, a GUI will be launched which can be used to send commands to the camera. The GUI itself will not be addressed in this document.

### **6.3.5 Take Camera Offline (Use Case)**

Operators with the proper functional rights may take a camera offline. A camera that has been taken offline may not be displayed or controlled until it is put back online.

### **6.3.6 Terminate Camera Control (Use Case)**

An operator with the proper functional rights may manually terminate a camera control session that the operator is actively using. Note that an operator who has the proper functional rights to establish the control session will always have the proper functional rights to terminate that session. Also, a camera control session may be terminated if that session is overridden by an appropriately privileged operator. Also, an active control session may be terminated if a camera is taken offline or if the camera is no longer displayed on a monitor within the controlling operator's monitor group as a result of displaying a NoVideoAvailable source. Note that part of this process will include terminating the camera control GUI, although that is beyond the scope of this document.

### **6.3.7 View Camera Details for Maint (Use Case)**

The details for Camera devices shall be available for viewing for the purpose of performing maintenance, as specified in the View Device Details For Maint use case.

### **6.3.8 View Camera List for Maint (Use Case)**

The system shall allow the user to view a list of Cameras that are candidates for device maintenance as specified in the View Device List for Maint use case.

### **6.3.9 View Device Details For Maint (Use Case)**

The system shall allow the details for a device to be viewed for the purpose of performing maintenance. This shall be supported for DMS, HAR, SHAZAM, TSS, and Camera devices. The details shall include the device type, name, and location left justified near the top of the page. All other information on the page shall also be left justified. Any actions that apply to the device in its current mode and based on the user's rights shall be available toward the top of the page, under the device type/name/location. The list of actions available for a device shall match the actions available for the device within the standard GUI, except as follows: The list of actions available shall exclude the ability to copy the device. The list of actions available shall exclude and the ability to remove the device. Exceptions specific to a device type may also apply and are specified in the extending use cases where applicable. The data displayed for a device shall match the data displayed for the device if the user were to view details for the device for other purposes (non maintenance activities) except as specified in extending use cases.

### **6.3.10 View Device List For Maint (Use Case)**

The system shall allow the user to view a list of devices that are candidates for maintenance. This shall include the ability to list DMS, HAR, SHAZAM, TSS, and Camera devices (in separate lists). Each list will identify the type of devices that are shown in the list. Each list can include all CHART devices of the selected type, or can be pre-filtered. The pre-filtering can be done by device mode/status or using the GUI's folder feature. When the folder feature is used, only devices that exist in folders that are tagged with the user's operations center are shown. If no such devices exist then all CHART devices of the specified type are shown (unfiltered list). The list of devices will show the number of devices that appear in the list. If the device is filtered, the list will also show the number of devices that would appear if the list is unfiltered and will show the filter(s) in use. The user shall be able to remove all filtering from a list that is filtered. The list will show each device with an icon to identify the device type and mode/status, the device name, and the device location. Each list will provide access to a details page for each device shown.

## 6.4 ManageCamera (Use Case Diagram)

An operator will interact with cameras in a variety of ways. Cameras may be taken online or offline. Monitors may be taken online or offline as well. Cameras may be displayed. Cameras may also be controlled. Note that the term control as it applies to cameras has a slightly different meaning than when applied other types of CHART devices, such as DMSs. An operator who controls a camera establishes a control session which typically lasts some number of minutes. During this control session, the operator sends multiple commands to the camera (e.g., Pan Left, Pan Stop). While the session is active, no other operator may send commands to the camera. An operator may also view which cameras are displayed on which monitors. Also, a camera may be revoked for display or control.

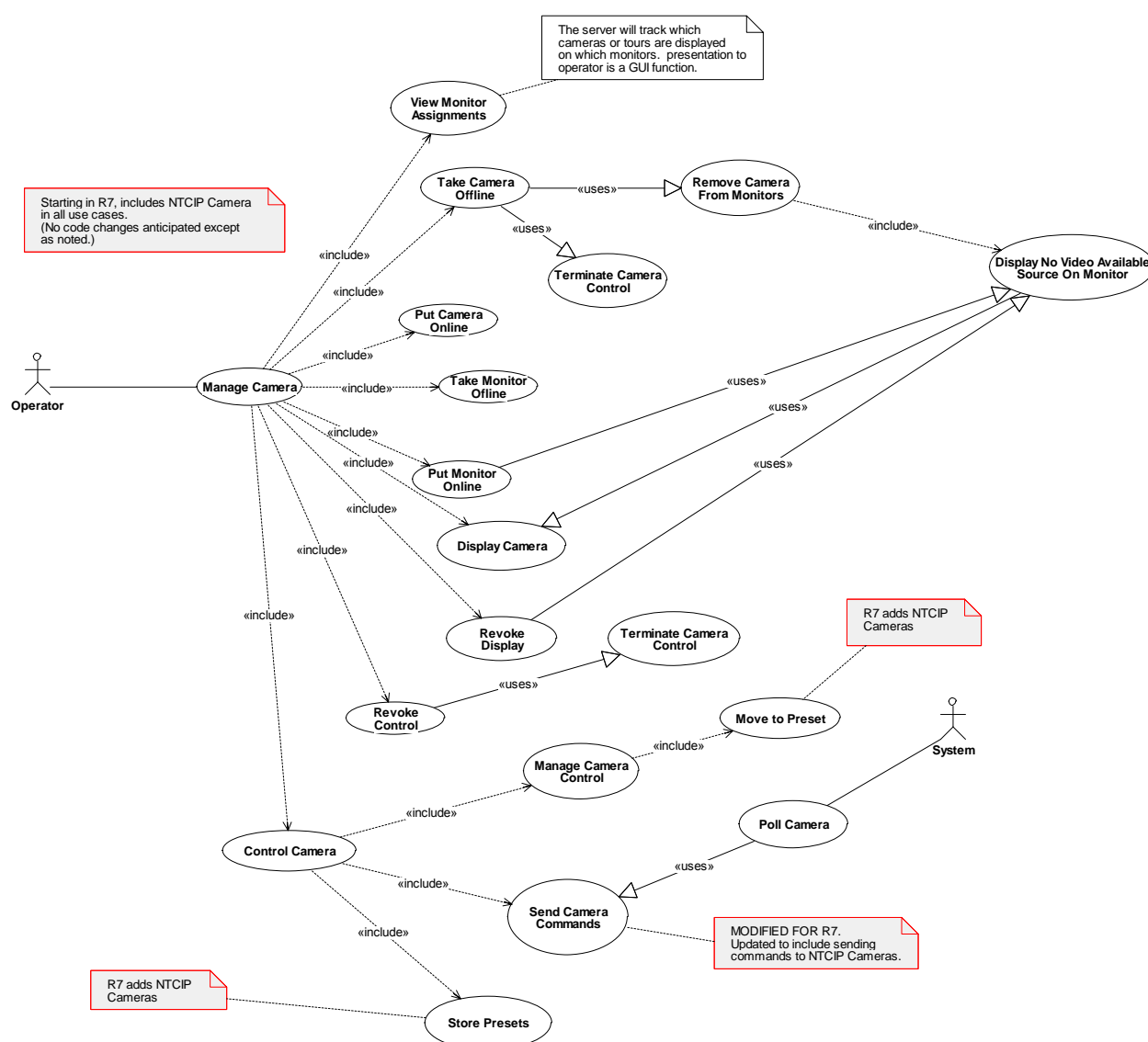


Figure 6-4ManageCamera (Use Case Diagram)

### **6.4.1 Control Camera (Use Case)**

An operator with the proper functional rights may control a camera. In this case, control refers to issuing commands to the camera to cause the camera to pan/tilt/zoom etc. Refer to the send Camera Commands use case diagram for more details.

### **6.4.2 Display Camera (Use Case)**

An operator with the correct functional rights may display a camera on a monitor. See the Display Camera use case diagram for a more detailed explanation.

### **6.4.3 Display No Video Available Source On Monitor (Use Case)**

A No Video Source will be displayed on a monitor when the camera image has been removed without being replaced by a new camera image. A No Video Source acts essentially like another camera in the system.

### **6.4.4 Manage Camera (Use Case)**

An operator with the correct functional rights may perform basic operations on a camera. Please refer to the Manage Camera Use Case diagram for more detailed information.

### **6.4.5 Manage Camera Control (Use Case)**

An operator with the proper functional rights may either request control of a camera or terminate control of a camera. If the camera control request is successful, a camera control session is established. See the Request Camera Control use case for further details. When camera control is terminated, the camera control session is terminated. See the Terminate Camera Control use case for further details.

### **6.4.6 Move to Preset (Use Case)**

A user with control of a camera shall be able to move the camera to a preset position stored with the camera. The Move to Preset Use Case can also be invoked by a camera tour with a preset associated with a camera.

### **6.4.7 Poll Camera (Use Case)**

A camera is polled by the system in order to establish the status of the camera control communications path. The polling process consists of sending the camera a poll command and receiving a response from the camera. This is done by the system for all cameras that are online, regardless of whether the cameras are controlled or not. It is also done immediately after camera control has been granted so that the camera control status is current. In addition, polling takes place while a camera is actively controlled. When a

camera is actively controlled, the polling is typically much more frequent than when the camera is not actively controlled.

#### **6.4.8 Put Camera Online (Use Case)**

An operator with the proper functional rights can put a camera online if the camera is currently offline. Putting the camera online makes it available for display and control to any operators having the proper functional rights.

#### **6.4.9 Put Monitor Online (Use Case)**

An operator with the proper functional rights can put a monitor online if the monitor is currently offline. Putting the monitor online makes it available for display to any operators having the proper functional rights.

#### **6.4.10 Remove Camera From Monitors (Use Case)**

When a camera has been taken offline, the camera image must be removed from any monitors on which it is displayed.

#### **6.4.11 Revoke Control (Use Case)**

An operator who revokes control of a camera does so from specific organizations. Since operators themselves are not “owned” by an organization, the organization of the camera control session is determined by the operators chosen (or assigned) monitor group.

#### **6.4.12 Revoke Display (Use Case)**

An operator who revokes display of a camera does so for a specific organization's monitor(s). This can include multiple organizations. This means that monitors owned by that organization cannot have the revoked camera image displayed on them. This includes a specific capability to block from the public (meaning monitors designated as public will be revoked).

#### **6.4.13 Send Camera Commands (Use Case)**

An operator with the proper functional rights may send commands to a camera. This includes sending the command to the camera and receiving a response from the camera. Commands sent to the camera include pan, tilt, zoom, iris control, focus, save preset, move to preset, and color balance control commands. Commands may also include camera reset, camera power, and camera titling commands.

#### **6.4.14 Store Presets (Use Case)**

A user with control of a camera shall be able to store a camera location (PTZ position) as a camera preset. Up to ten presets can be stored per camera. Camera position and title are



stored in the camera for efficiency, as well as in the CHART database.

#### **6.4.15 Take Camera Offline (Use Case)**

Operators with the proper functional rights may take a camera offline. A camera that has been taken offline may not be displayed or controlled until it is put back online.

#### **6.4.16 Take Monitor Offline (Use Case)**

Operators with the proper functional rights may take a monitor offline. A monitor that has been taken offline may not be displayed on until it is put back online

#### **6.4.17 Terminate Camera Control (Use Case)**

An operator with the proper functional rights may manually terminate a camera control session that the operator is actively using. Note that an operator who has the proper functional rights to establish the control session will always have the proper functional rights to terminate that session. Also, a camera control session may be terminated if that session is overridden by an appropriately privileged operator. Also, an active control session may be terminated if a camera is taken offline or if the camera is no longer displayed on a monitor within the controlling operator's monitor group as a result of displaying a NoVideoAvailable source. Note that part of this process will include terminating the camera control GUI, although that is beyond the scope of this document.

#### **6.4.18 View Monitor Assignments (Use Case)**

An operator may view which cameras or camera tours are assigned to which monitors. This information will be made available by the server for the GUI to interpret. The presentation to the user is beyond the scope of this design.

## 6.5 ManageCameraControl (Use Case Diagram)

An operator may control establish a camera control session which will allow the operator to issue Pan/Tilt/Zoom and other commands to the camera while that control session is active. Only one camera control session at a time will be active so that only one operator at a time may control a particular camera. The cameras that are available for an operator to control include only those cameras that are displayed on monitors that are within the operator's monitor group. Presumably, the monitors are physically visible to the operator. This is so that the operator will be able to see the camera image while the camera is actually being moved. In addition an operator may be able to override an existing camera control session, thereby taking control of a camera from another operator. The specific business rules which govern camera control override are described in the Override Camera Control use case.

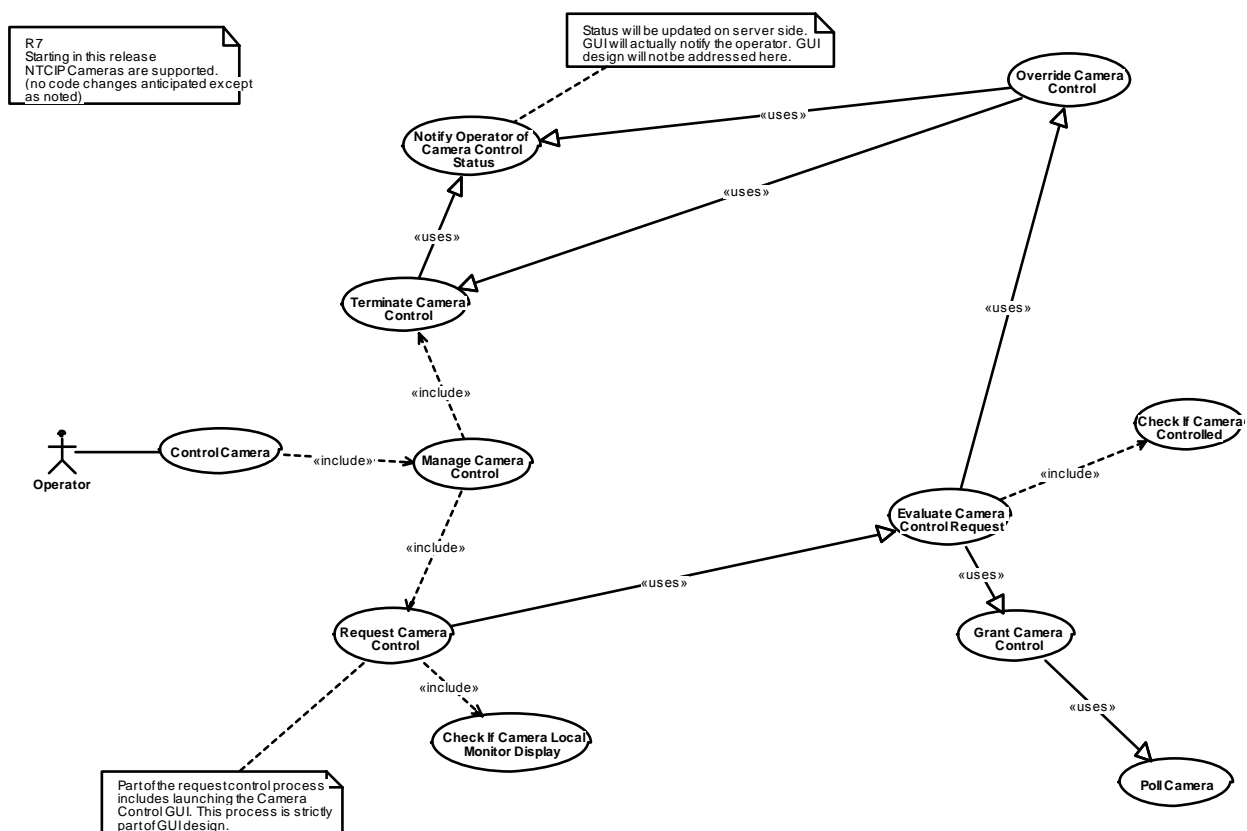


Figure 6-5 ManageCameraControl (Use Case Diagram)

### **6.5.1 Check If Camera Controlled (Use Case)**

In order to evaluate a camera control request, the system must determine whether the camera is currently controlled by another operator.

### **6.5.2 Check If Camera Local Monitor Display (Use Case)**

A camera control request will only be granted if the camera is displayed on a monitor that is in the monitor group of the operator requesting control. This is to implement the requirement that an operator may only control a camera when the operator can actually see the camera.

### **6.5.3 Control Camera (Use Case)**

An operator with the proper functional rights may control a camera. In this case, control refers to issuing commands to the camera to cause the camera to pan/tilt/zoom etc. Refer to the send Camera Commands use case diagram for more details.

### **6.5.4 Evaluate Camera Control Request (Use Case)**

An operator may control establish a camera control session which will allow the operator to issue Pan/Tilt/Zoom and other commands to the camera while that control session is active. Only one camera control session at a time will be active so that only one operator at a time may control a particular camera. The cameras that are available for an operator to control include only those cameras that are displayed on monitors that are within the operator's monitor group. Presumably, the monitors are physically visible to the operator. This is so that the operator will be able to see the camera image while the camera is actually being moved. In addition an operator may be able to override an existing camera control session, thereby taking control of a camera from another operator. The specific business rules which govern camera control override are described in the Override Camera Control use case.

### **6.5.5 Grant Camera Control (Use Case)**

When a camera control request has been granted, the "circuit" is established, the control session becomes fully active, and the camera is polled so that the camera status may be immediately updated.

### **6.5.6 Manage Camera Control (Use Case)**

An operator with the proper functional rights may either request control of a camera or terminate control of a camera. If the camera control request is successful, a camera control session is established. See the Request Camera Control use case for further details. When camera control is terminated, the camera control session is terminated. See the Terminate

Camera Control use case for further details.

### **6.5.7 Notify Operator of Camera Control Status (Use Case)**

An operator will be notified of camera control status under a number of circumstances. If another operator overrides the controlling operator's camera control session, the controlling operator will be notified. If an administrator with sufficient privileges takes a camera offline, then the controlling operator will be notified. Also, if the controlled camera is no longer displayed on a monitor within the controlling operator's monitor group, the controlling operator will be notified that their camera control session has been terminated. The actual mechanism used to notify the operator through the GUI is beyond the scope of the server side design.

### **6.5.8 Override Camera Control (Use Case)**

The Override Camera Control use case is invoked when an operator with the proper functional rights requests control of a camera that is currently controlled by another operator but where control would otherwise be allowed. If, based on a set of business rules, the operator may override the existing camera control session, the requesting operator will have the option to override the existing camera control session. If the requesting operator chooses to override, the existing control session will be terminated and the new one will start. Note that if the operator does choose to override an existing control session, control may not be granted immediately. This is because the existing camera control session will not terminate until all long running commands, such as setting a title on certain types of cameras, have completed. A requesting operator may override an existing camera control session based when the requesting operator has the Override Camera Control functional right for the camera's owning organization

### **6.5.9 Poll Camera (Use Case)**

A camera is polled by the system in order to establish the status of the camera control communications path. The polling process consists of sending the camera a poll command and receiving a response from the camera. This is done by the system for all cameras that are online, regardless of whether the cameras are controlled or not. It is also done immediately after camera control has been granted so that the camera control status is current. In addition, polling takes place while a camera is actively controlled. When a camera is actively controlled, the polling is typically much more frequent than when the camera is not actively controlled.

### **6.5.10 Request Camera Control (Use Case)**

An operator with the proper functional rights may request control of a camera. This means that the operator may send pan/tilt/zoom (PTZ) and other commands to the camera. The system evaluates the request, and will accept the request, prompt the operator to override an existing camera control session, or reject the request. If the request is accepted or the user chooses to override an existing control session, a GUI will be launched which can be used

to send commands to the camera. The GUI itself will not be addressed in this document.

#### **6.5.11 Terminate Camera Control (Use Case)**

An operator with the proper functional rights may manually terminate a camera control session that the operator is actively using. Note that an operator who has the proper functional rights to establish the control session will always have the proper functional rights to terminate that session. Also, a camera control session may be terminated if that session is overridden by an appropriately privileged operator. Also, an active control session may be terminated if a camera is taken offline or if the camera is no longer displayed on a monitor within the controlling operator's monitor group as a result of displaying a NoVideoAvailable source. Note that part of this process will include terminating the camera control GUI, although that is beyond the scope of this document.

## 6.6 R7VerifyNTCIPCameraCompatibility (Use Case Diagram)

This diagram shows the use cases for the NTCIP Camera Compatibility Tester, a stand alone tool made available to Camera vendors to determine if their NTCIP camera is compatible with the CHART system.

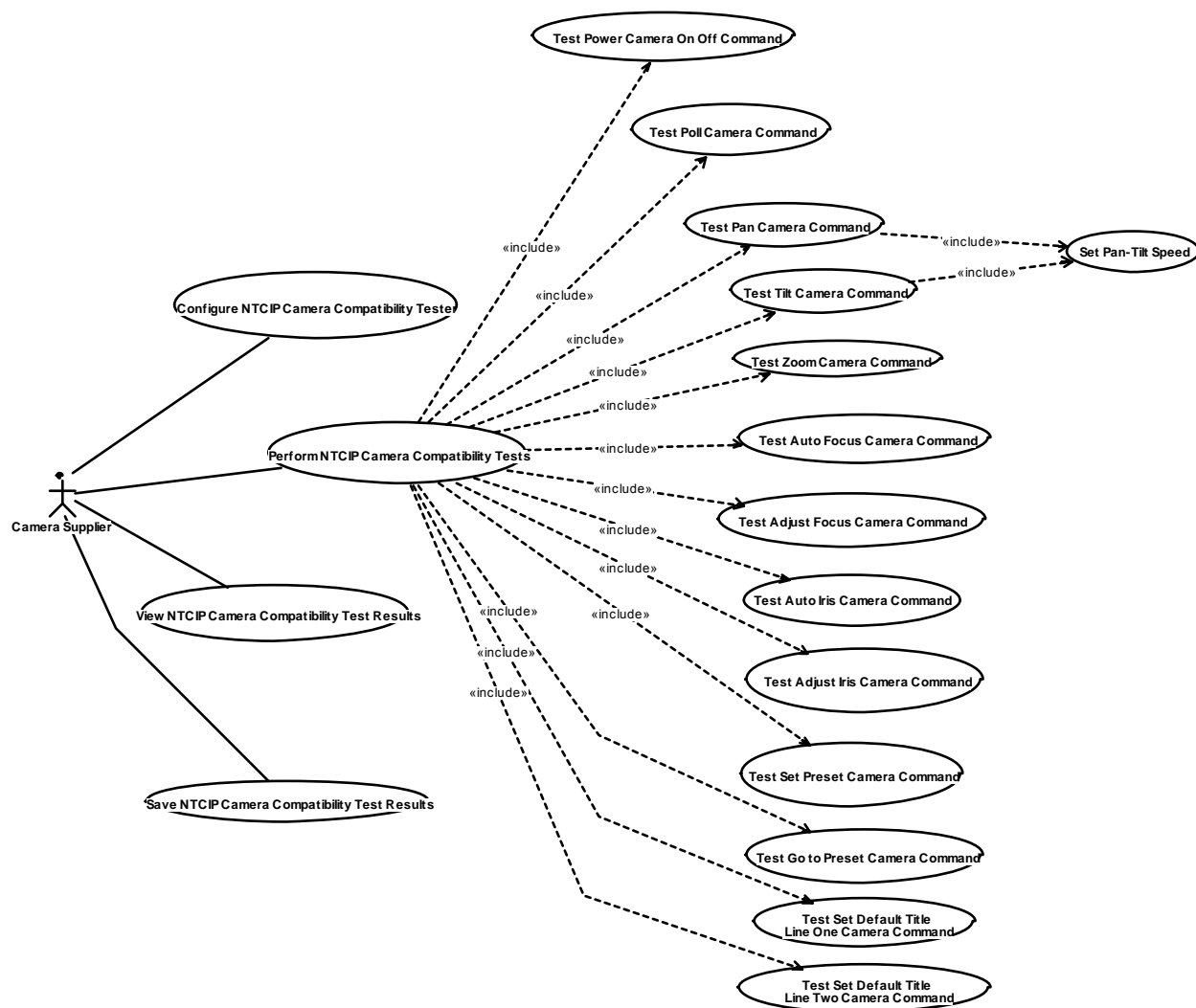


Figure 6-6 R7VerifyNTCIPCameraCompatibility (Use Case Diagram)

### **6.6.1 Configure NTCIP Camera Compatibility Tester (Use Case)**

This use case allows the user to configure settings in the camera tester to support testing.

### **6.6.2 Perform NTCIP Camera Compatibility Tests (Use Case)**

The NTCIPCompatibilityTest shall allow Camera suppliers to test if an NTCIP Camera is compatible with the CHART system.

### **6.6.3 Save NTCIP Camera Compatibility Test Results (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to save the test results to a file.

### **6.6.4 Set Pan-Tilt Speed (Use Case)**

The system will adjust camera pan/tilt speed based on the current zoom level and configured minimum and maximum pan/tilt speeds.

### **6.6.5 Test Adjust Focus Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Adjust Focus feature operates properly on an NTCIP Camera.

### **6.6.6 Test Adjust Iris Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Adjust Iris feature operates properly on an NTCIP Camera.

### **6.6.7 Test Auto Focus Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Auto Focus feature operates properly on an NTCIP Camera.

### **6.6.8 Test Auto Iris Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Auto Iris feature operates properly on an NTCIP Camera.

### **6.6.9 Test Go to Preset Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Go to Preset feature operates properly on an NTCIP Camera.

#### **6.6.10 Test Pan Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Pan feature operates properly on an NTCIP Camera.

#### **6.6.11 Test Poll Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Poll Camera command operates properly

#### **6.6.12 Test Power Camera On Off Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Set Power On/Off feature operates properly on an NTCIP Camera.

#### **6.6.13 Test Set Default Title Line One Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Set Default Title Line one feature operates properly on an NTCIP Camera.

#### **6.6.14 Test Set Default Title Line Two Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Default Title line two operates properly on an NTCIP Camera.

#### **6.6.15 Test Set Preset Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Set Preset feature operates properly on a NTCIP Camera.

#### **6.6.16 Test Tilt Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Tilt feature operates properly on an NTCIP Camera.

#### **6.6.17 Test Zoom Camera Command (Use Case)**

The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Zoom feature operates properly on an NTCIP Camera.



## 6.7 SendCameraCommands (Use Case Diagram)

An operator with the proper functional rights may control a camera. Once a control session has been established, the operator will use the camera control GUI to issue control commands to the camera. Those commands include pan, tilt, zoom, iris, focus, color balance, camera reset, preset, and camera title commands. For each command sent to the camera, a response shall be received from the camera. In addition to commands sent by the user, the system will send poll commands to the camera and evaluate the responses from those poll commands. The Poll Camera use case is described as part of the Manage Camera use case diagram. Note that each type of command will have separate functional rights so that some operators may be able to send pan, tilt, and zoom commands to the camera but will not be allowed to set camera's color balance, for instance. Only COHU 3955, Surveyor VFT, and NTCIP cameras may be controlled. COHU 3955 and NTCIP commands will be sent to the camera through an IP based encoder, which will convert IP control commands to serial camera control commands which will be sent to the camera over the encoder's COM port. Surveyor VFT commands will be sent through either an IP based Encoder or a Command Processor which in turn sends over an RS-232 port. The Command Processor manages sending commands to multiple cameras attached to a single RS-232 port.

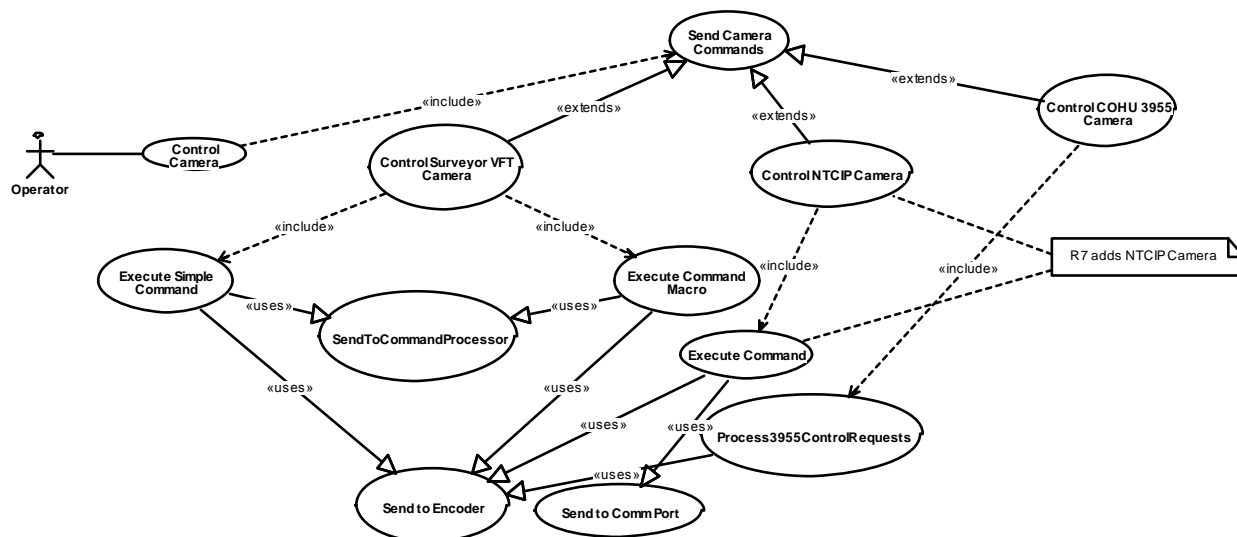


Figure 6-7 SendCameraCommands (Use Case Diagram)

### 6.7.1 Control Camera (Use Case)

An operator with the proper functional rights may control a camera. In this case, control refers to issuing commands to the camera to cause the camera to pan/tilt/zoom etc. Refer to the send Camera Commands use case diagram for more details.

### **6.7.2 Control COHU 3955 Camera (Use Case)**

The Control COHU camera use case provides the functionality needed to send commands to COHU 3955 cameras. The command will be built, meaning that the command bytes will be generated, and those commands will be sent to the camera. Each command sent to the camera should elicit a response. Should no response be received from the camera, an error shall be returned to the user indicating that the command has failed. However, the camera will remain online and available to continue to receive commands.

### **6.7.3 Control NTCIP Camera (Use Case)**

The NTCIP camera use case provides the functionality needed to send commands to NTCIP cameras. The command will be built, meaning that the command bytes will be generated, and those commands will be sent to the camera. Each command sent to the camera should elicit a response. Should no response be received from the camera, an error shall be returned to the user indicating that the command has failed. However, the camera will remain online and available to continue to receive commands.

### **6.7.4 Control Surveyor VFT Camera (Use Case)**

The Control Surveyor VFT camera use case provides the functionality needed to send commands to Surveyor VFT cameras. The command will be built, meaning that the command bytes will be generated, and those commands will be sent to the camera. Each command sent to the camera should elicit a response. Should no response be received from the camera, an error shall be returned to the user indicating that the command has failed. However, the camera will remain online and available to continue to receive commands.

### **6.7.5 Execute Command (Use Case)**

For the NTCIP camera, certain commands (such as setting a camera title) require multiple commands to be sent to the camera. Camera control request processing for the NTCIP camera includes building the binary command, sending that command to the camera via a CameraControlComPort or IP based Encoder, and receiving a response from the camera via the CameraControlComPort or IP based Encoder.

### **6.7.6 Execute Command Macro (Use Case)**

For the Surveyor VFT camera, certain commands (such as setting a camera title) require multiple commands to be generated from a compiled macro. Once these simple commands have been assembled, camera control request processing for the Surveyor VFT camera includes actually building the simple binary command, sending that command to the camera via a Command Processor or IP based Encoder and receiving a response from the camera via the Command Processor or IP based Encoder.

### **6.7.7 Execute Simple Command (Use Case)**

Simple camera control request processing for the Surveyor VFT camera includes actually building the simple binary command, sending that command to the camera via either a Command Processor or an IP based Encoder and receiving a response from the camera via the Command Processor or IP based Encoder.

### **6.7.8 Process3955ControlRequests (Use Case)**

Camera control request processing for the COHU 3955 includes actually building the binary command, sending that command to the camera via an IP sending device and receiving a response from the camera via an IP sending device. Although the camera itself sends and receives data using an RS-422 connection, CHART instead communicates with an IP based encoder that converts TCP/IP data to serial data for transmission to the camera, and converts serial data from the camera to TCP/IP for transmission to the system.

### **6.7.9 Send Camera Commands (Use Case)**

An operator with the proper functional rights may send commands to a camera. This includes sending the command to the camera and receiving a response from the camera. Commands sent to the camera include pan, tilt, zoom, iris control, focus, save preset, move to preset, and color balance control commands. Commands may also include camera reset, camera power, and camera titling commands.

### **6.7.10 Send to Comm Port (Use Case)**

A comm port is used for control of NTCIP cameras from the tester. Commands are sent to/and received from the comm port. The comm port takes communicates with the camera via RS-232.

### **6.7.11 Send to Encoder (Use Case)**

An IP based Encoder can be used for control of COHU 3955, Vicon Surveyor VFT, or NTCIP cameras. Commands are sent to/and received from the Encoder. The Encoder itself, takes communicates with the camera via RS-232 or RS-422.

### **6.7.12 SendToCommandProcessor (Use Case)**

The utilize command processor use case encompasses sending and receiving camera commands to/from multiple cameras attached to a single RS-232 port.

## 6.8 View NTCIP Camera Compatibility Test Results (Use Case)

The NTCIP Camera Compatibility Tester shall provide output that shows the user the results of the tests that are run.

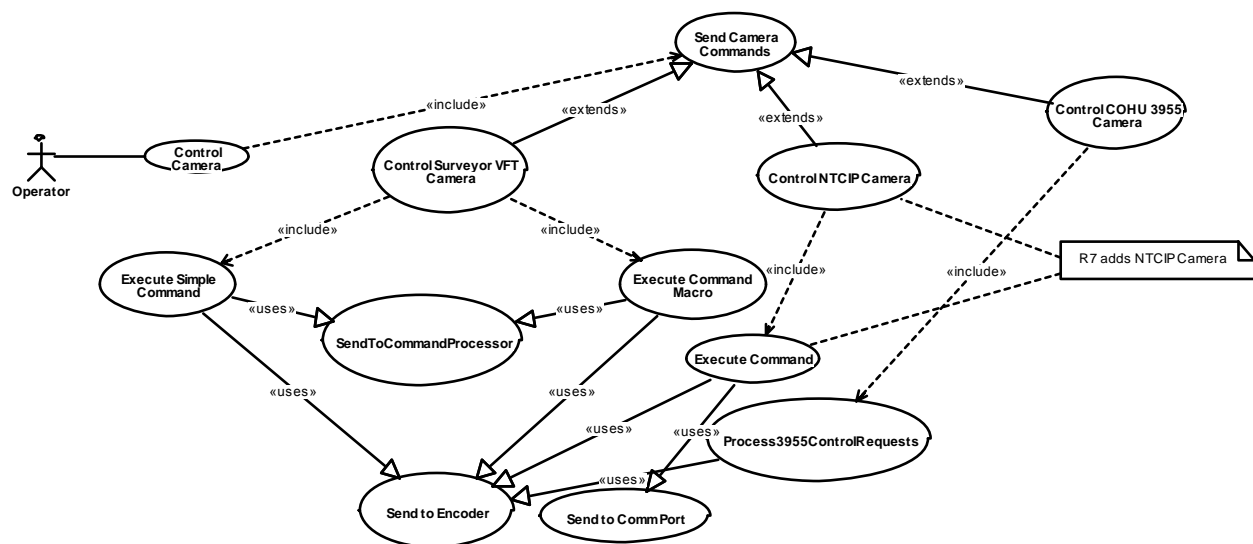


Figure 6-8 SendCameraCommands (Use Case Diagram)

### 6.8.1 Control Camera (Use Case)

An operator with the proper functional rights may control a camera. In this case, control refers to issuing commands to the camera to cause the camera to pan/tilt/zoom etc. Refer to the send Camera Commands use case diagram for more details.

### 6.8.2 Control COHU 3955 Camera (Use Case)

The Control COHU camera use case provides the functionality needed to send commands to COHU 3955 cameras. The command will be built, meaning that the command bytes will be generated, and those commands will be sent to the camera. Each command sent to the camera should elicit a response. Should no response be received from the camera, an error shall be returned to the user indicating that the command has failed. However, the camera will remain online and available to continue to receive commands.

### 6.8.3 Control NTCIP Camera (Use Case)

The NTCIP camera use case provides the functionality needed to send commands to NTCIP cameras. The command will be built, meaning that the command bytes will be generated, and those commands will be sent to the camera. Each command sent to the camera should

elicit a response. Should no response be received from the camera, an error shall be returned to the user indicating that the command has failed. However, the camera will remain online and available to continue to receive commands.

#### **6.8.4 Control Surveyor VFT Camera (Use Case)**

The Control Surveyor VFT camera use case provides the functionality needed to send commands to Surveyor VFT cameras. The command will be built, meaning that the command bytes will be generated, and those commands will be sent to the camera. Each command sent to the camera should elicit a response. Should no response be received from the camera, an error shall be returned to the user indicating that the command has failed. However, the camera will remain online and available to continue to receive commands.

#### **6.8.5 Execute Command (Use Case)**

For the NTCIP camera, certain commands (such as setting a camera title) require multiple commands to be sent to the camera. Camera control request processing for the NTCIP camera includes building the binary command, sending that command to the camera via a CameraControlComPort or IP based Encoder, and receiving a response from the camera via the CameraControlComPort or IP based Encoder.

#### **6.8.6 Execute Command Macro (Use Case)**

For the Surveyor VFT camera, certain commands (such as setting a camera title) require multiple commands to be generated from a compiled macro. Once these simple commands have been assembled, camera control request processing for the Surveyor VFT camera includes actually building the simple binary command, sending that command to the camera via a Command Processor or IP based Encoder and receiving a response from the camera via the Command Processor or IP based Encoder.

#### **6.8.7 Execute Simple Command (Use Case)**

Simple camera control request processing for the Surveyor VFT camera includes actually building the simple binary command, sending that command to the camera via either a Command Processor or an IP based Encoder and receiving a response from the camera via the Command Processor or IP based Encoder.

#### **6.8.8 Operator (Actor)**

An operator is a user of the system who has been assigned a valid username/password combination and granted roles for system access.

#### **6.8.9 Process3955ControlRequests (Use Case)**

Camera control request processing for the COHU 3955 includes actually building the binary command, sending that command to the camera via an IP sending device and

receiving a response from the camera via an IP sending device. Although the camera itself sends and receives data using an RS-422 connection, CHART instead communicates with an IP based encoder that converts TCP/IP data to serial data for transmission to the camera, and converts serial data from the camera to TCP/IP for transmission to the system.

#### **6.8.10 Send Camera Commands (Use Case)**

An operator with the proper functional rights may send commands to a camera. This includes sending the command to the camera and receiving a response from the camera. Commands sent to the camera include pan, tilt, zoom, iris control, focus, save preset, move to preset, and color balance control commands. Commands may also include camera reset, camera power, and camera titling commands.

#### **6.8.11 Send to Comm Port (Use Case)**

A comm port is used for control of NTCIP cameras from the tester. Commands are sent to/and received from the comm port. The comm port takes communicates with the camera via RS-232.

#### **6.8.12 Send to Encoder (Use Case)**

An IP based Encoder is used for control of either COHU 3955, Vicon Surveyor VFT, or NTCIP cameras. Commands are sent to/and received from the Encoder. The Encoder itself, takes communicates with the camera via RS-232 or RS-422.

#### **6.8.13 SendToCommandProcessor (Use Case)**

The utilize command processor use case encompasses sending and receiving camera commands to/from multiple cameras attached to a single RS-232 port.

## 7 Detailed Design – NTCIP Camera

### 7.1 Human-Machine Interface

#### 7.1.1 Add/Edit NTCIP Camera

The existing screens used to add a camera to the system or to edit an existing camera are changed to support the NTCIP camera model. When adding a new camera to the system, a new model choice is available, NTCIP. See the Camera Model field in the screen shot below:

**Add Camera**

Camera Model:  New!

Name:

Owning Organization:

Maintaining Organization:

Sending Device(s):

Encoder(s) [Add](#)  
Switch(s) [Add](#)

When Camera Model is set to NTCIP, additional configuration fields appear that are specific to the NTCIP Camera Model. This includes the NTCIP Community String, HDLC Framing option, and speeds for each type of camera movement (pan, tilt, zoom, focus). The screen capture below provides an example of how these configuration options will appear. Note that the speeds for pan and tilt will actually include two speeds, a min and max, to support variable speed pan/tilt. Two additional fields will also be included to specify the min and max zoom values supported by the camera. These two fields are also required for variable speed pan/tilt and not shown.

Control Device: ☒ IP via CODEC ☐ Command Processor ☐ Comm Port

Model Type: ☒ CORETEC MPEG4 ☐ iMPath MPEG2

Hostname / IP:

Port:

NTCIP Community:  New!

NTCIP HDLC Framing Required: ☐

NTCIP Variable Control Pan Speed:

NTCIP Variable Control Tilt Speed:

NTCIP Variable Control Zoom Speed:

NTCIP Variable Control Focus Speed:

**COMM PORT SETTINGS FOR INFORMATIONAL PURPOSES ONLY:**

Baud: ☐ 300 ☐ 600 ☐ 1200 ☐ 2400 ☐ 4800 ☒ 9600 ☐ 19200

☐ 28800 ☐ 38400 ☐ 57600 ☐ 115200

Data Bits: ☐ 5 ☐ 6 ☐ 7 ☒ 8

Parity: ☒ None ☐ Space ☐ Mark ☐ Even ☐ Odd

Stop Bits: ☒ 1 ☐ 1.5 ☐ 2

Flow Control: ☒ None ☐ XON/XOFF ☐ RTS/CTS

Polling Enabled Outside Of Control Session: ☐ No ☒ Yes

Poll Interval (seconds):

Poll Interval Within Control Session (seconds):

Device Logging Enabled: ☒ No ☐ Yes

Default Title(s):

Line 1:

Line 2:

## 7.1.2 View NTCIP Camera Details

The Camera Details page is changed to show fields specific to NTCIP Cameras when the Camera model is set to NTCIP. The fields included are those discussed above regarding Add/Edit Camera and an example is shown below. Note that additional fields will also appear on this screen that are required for variable speed pan/tilt, such as min/max pan and tilt speeds (instead of a single speed for each) and min/max zoom values.

### Control Communication Settings:

Control Device Type:	IP
NTCIP Community String	public
NTCIP HDLC Framing:	ENABLED
Codec Hostname / IP Address:	170.90.22.15
Model Type:	CoreTec MPEG 4
Codec Port:	5000
Camera Baud:	9600
Camera Data Bits:	8
Camera Parity:	None
Camera Stop Bits:	1
Camera Flow Control:	None

New!

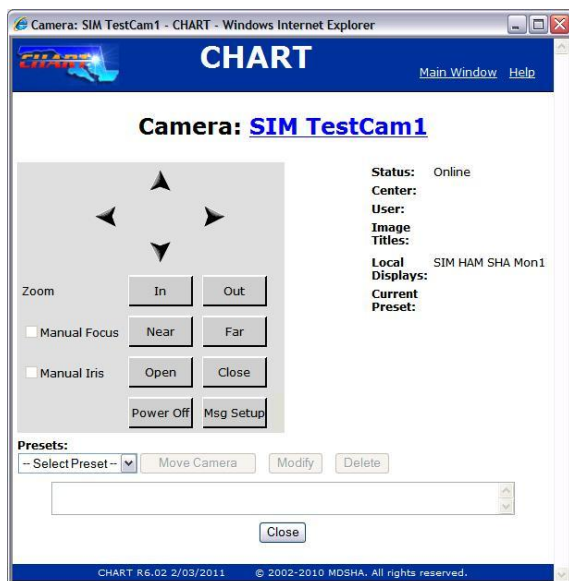
### Controllable Camera Settings:

Polling Interval	4 seconds
In Control Session:	
Polling Interval	300 seconds
Outside of Control Session:	
Device Logging:	OFF
Default Title Line 1:	TITLE LINE 1
Default Title Line 2:	title line 2
NTCIP Variable Control Pan Speed:	127
NTCIP Variable Control Tilt Speed:	127
NTCIP Variable Control Zoom Speed:	127
NTCIP Variable Control Focus Speed:	127

New!

## 7.1.3 Control NTCIP Camera

The camera control dialog for NTCIP cameras will operate like the control dialogs that currently exist for Vicon and Cohu cameras. Some features of the Vicon and Cohu cameras are not supported by the NTCIP camera model and those features will not appear on the NTCIP camera control dialog, however the features that do exist will operate identically to the Vicon and Cohu versions of these features. See the sample NTCIP camera control dialog below:



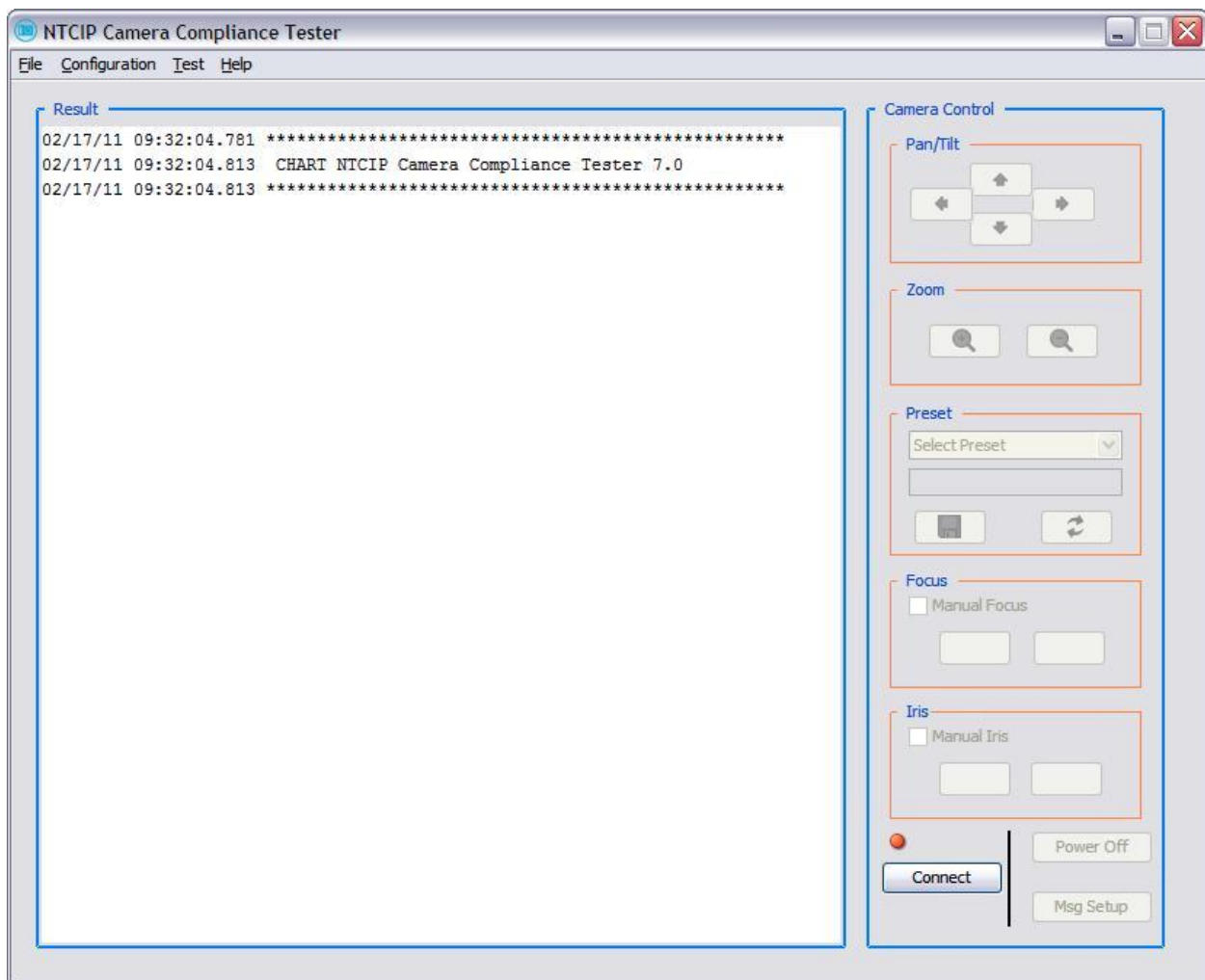


The following features are supported on the NTCIP Camera control dialog:

Pan, tilt, zoom, focus (in/out), auto focus, iris (open/close), auto iris, set titles (2 lines), power off, power on, set preset, move to preset.

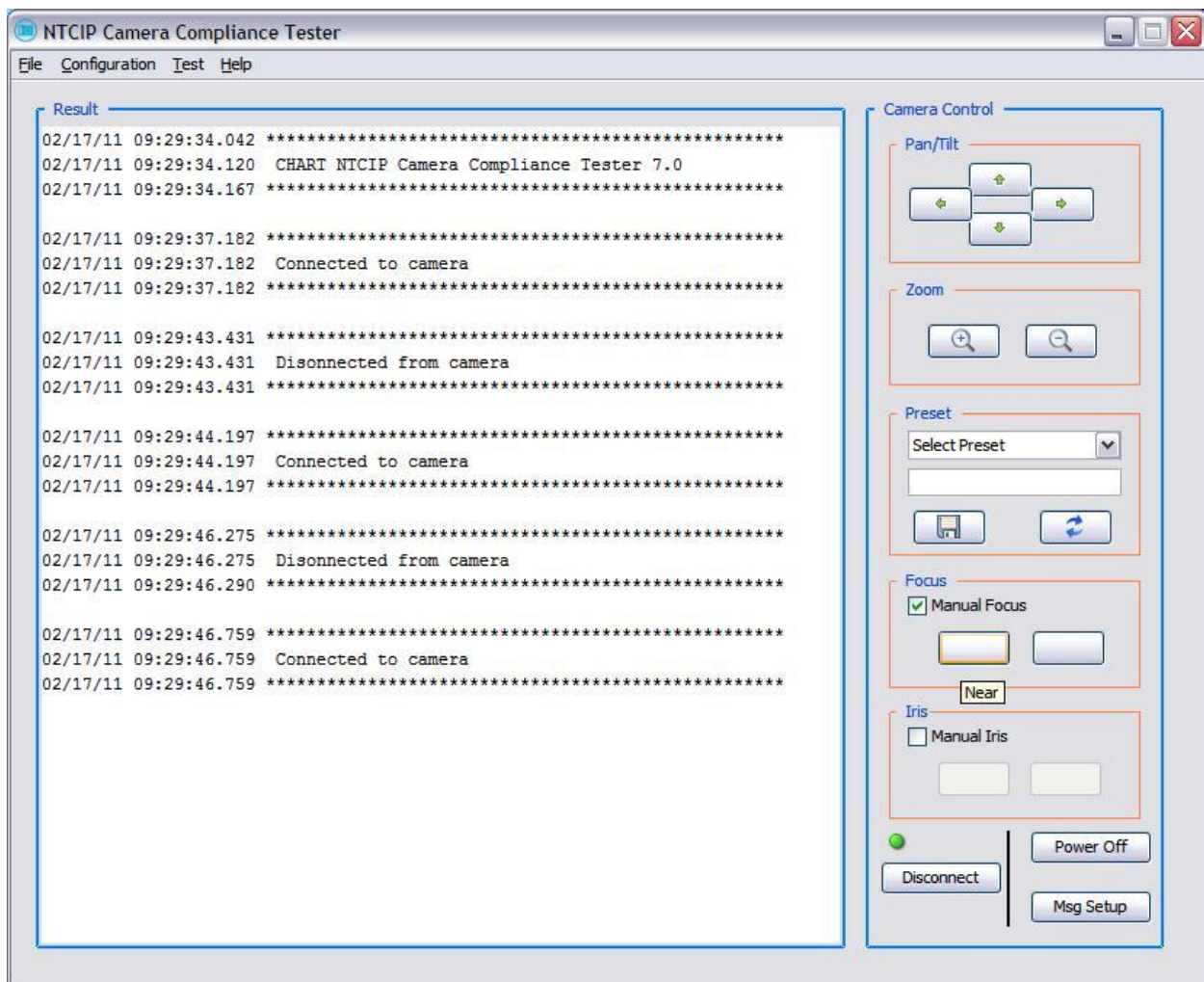
#### 7.1.4 NTCIP Camera Compliance Tester

The NTCIP Camera Compliance Tester is a stand-alone application that includes its own installer, separate from the CHART system. The main window of the application will, contain a text area where informational messages appear, a menu system, and a camera control area. A mock up of this screen is shown below.



**Figure 7-1. NTCIP Camera Compliance Tester (Main Screen)**

After comms settings are configured and the Connect button is hit, once the camera connection is established, the Connect light turns green and all control buttons become active, as shown below.



**Figure 7-2 NTCIP Camera Compliance Tester (Connecting)**

## Result Area

The result area contains information about the tests as they are run. The text in this area can be saved to a file after testing to allow the vendor to provide their testing results to SHA. The area can also be cleared if desired to start a fresh set of results.

## Camera Control Area

The Camera Control area of the main window is used to send control commands to the camera, such as Pan, Tilt, Zoom, and Focus. Presets can also be saved and used from this area. Note that the screen mock up may change in appearance, but is used to convey the idea that controls for these features will exist in a separate area of the main window.

## File Menu

The File Menu provides the ability to save the results shown in the output area, or to clear the output area. A menu item also exists to allow the user to exit the application.

## Configuration Menu

The configuration menu provides access to the communications configuration dialog. This dialog allows the user to configure settings related to communications with and manipulation of the camera being tested. The tester application supports two types of communications; direct connect RS-232 or TCP/IP. A mock up of the communications configuration dialog is shown below for TCP/IP and RS-232 communications:

The image displays two side-by-side mockups of the 'Communication Settings' dialog box. The left mockup shows the 'TCP/IP' tab selected, with fields for IP Address (127.0.0.1) and Port (8000). The right mockup shows the 'RS232' tab selected, with fields for Comm Port Name (COM1), Baud Rate (9600), Data Bits (8), Parity (None), Stop Bits (1), and Flow Control (None). Both mockups also include a 'General Settings' section with fields for Drop Address, SNMP Community, HDLC Frame Required, Receive, Initial Timeout (ms), Minimum Pan Speed, Maximum Pan Speed, Minimum Tilt Speed, Maximum Tilt Speed, Zoom Speed, Focus Speed, Manual Focus, Manual Iris, Minimum Zoom Position, and Maximum Zoom Position. Each mockup has 'Save Settings' and 'Cancel' buttons at the bottom.

**Figure 7-3 NTCIP Camera Compliance Tester (Configuration)**

## Tests Menu

The tests menu provides access to tests that can be run that do not require camera movement and instead result in status being added to the output area. An example is for the Poll command, which causes the application to poll the camera and display the poll results in the output area.

## Help Menu

The help menu provides access to version information and a help page.

## 7.2 System Interfaces

The class diagrams in this section describe the CORBA interface classes and relationships that are being added or modified to support the NTCIP Camera feature.

### 7.2.1 Class Diagrams

#### 7.2.1.1 VideoHighLevel (Class Diagram)

This diagram shows the High Level CHART II CORBA interfaces. This diagram does not show all VideoService IDL elements, but shows the highest level elements and their interrelationships. For further details, see VideoHighLevel-VideoSource, VideoHighLevel-VideoSink, and VideoHighLevel-VideoTransmission diagrams. The collection of these last three diagrams show all planned CORBA/IDL interface objects for the CHART II Video Service. In all four of these diagrams, some boxes are shown indicating objects planned to be implemented for later releases. These objects have been considered for future planning purposes, to ensure that the current design is well-thought out enough to be able to accommodate future planned enhancements.

This diagram shows cameras and related information generally on the left side, monitors and related information generally on the right side, and video transmission and routing capabilities in the central part of the diagram. The VideoProvider interface is the top of the interface set which contains the VideoCamera interface. VideoSource includes video sources including fixed cameras, image generators, etc. Likewise on the right side, VideoCollector is at the top, opposite VideoProvider, with VideoSink and Monitor lower down. In addition to VideoSource and VideoSink objects, BridgeCircuit objects will also be VideoProviders and VideoCollectors, since any bridge circuit both collects video from some other VideoProvider and provides video to the next VideoCollector in line. Multiple bridge circuits may be present between the ultimate VideoProvider (i.e., the VideoSource, that is, the camera, the true source of the image) and the ultimate VideoCollector (i.e., the VideoSink, that is, the monitor, the final sink of the image).



#### **7.2.1.1.3 CameraControlComPort (Class)**

The CameraControlComPort interface is implemented by a class representing a COM port with direct connection to the control port of a video camera. It is used to send video camera control commands and return responses to a camera control process.

#### **7.2.1.1.4 CameraControlDevice (Class)**

The CameraControlDevice interface is implemented by classes which provides communications for access to control functions for a video camera. This includes encoders and direct COM ports.

#### **7.2.1.1.5 Codec (Class)**

The Codec interface is implemented by objects representing codec devices (that is, encoders and decoders). It defines generic methods to be implemented by both encoders and decoders.

#### **7.2.1.1.6 COHU3955Camera (Class)**

The COHU3955Camera interface is implemented by objects representing COHU model 3055 video cameras. It extends the ControllableVideoCamera interface by adding methods unique to the COHU 3955 cameras (unique within the universe of camera types planned for implementation within CHART II).

#### **7.2.1.1.7 CommandProcessor (Class)**

The CommandProcessor class provides an implementation of the CommandProcessor interface and is derived from the CameraControlDevice class. The CommandProcessor manages the control of multiple cameras attached to one or more COM ports. The CommandProcessor may or may not be local to the camera that is being controlled.

#### **7.2.1.1.8 CommEnabled (Class)**

The CommEnabled interface is implemented by objects that can have their communications turned on or off. This interface also supports a maintenance mode (although any given implementation may choose to implement putInMaintenanceMode() by throwing a CHART2Exception, if maintenance mode is not supported by that particular implementation). This interface is typically implemented only for field devices.

#### **7.2.1.1.9 ControllableVideoCamera (Class)**

The ControllableVideoCamera interface is implemented by objects representing controllable video cameras within the CHART II system. The ControllableVideoCamera interface represents a controllable video camera as opposed to an uncontrollable, immovable VideoCamera. Current plans call for classes to represent a COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera, and there are interfaces defined for each of these subtypes of ControllableVideoCamera. The ControllableVideoCamera interface includes all methods common to the two known types

of video cameras currently in use by MDSHA, although it is likely to contain a superset of methods which would be implemented by the entire universe of all video cameras which could someday be used. This interface may have to be refined in the event that future brands or models of video cameras might be incorporated under CHART II, but it is an appropriate set of methods for the present day. Current plans call for classes to represent a COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera.

#### **7.2.1.1.10 ControllingInfo (Class)**

The ControllingInfo structure contains information about the entity controlling (or requesting to control) a VideoCamera.

#### **7.2.1.1.11 Decoder (Class)**

The Decoder interface is implemented by classes representing any type of video decoder. The Decoder interface includes both the Codec and the VideoReceivingDevice interfaces.

#### **7.2.1.1.12 DiagonallyMovable (Class)**

The DiagonallyMovable interface is implemented by VideoCamera-enabled classes which can be moved diagonally in addition to standard orthogonal pan and tilt commands. A particular implementation may support 45-degree movements only, in which case the panDir and tiltDir parameters are +/- 1 to indicate direction only, or an implementation may support 360 degrees of motion, in which case, in addition to signs, the relative ratios of the parameters indicate the percent of movement proportionally in the pan/tilt directions. This interface is expected to be implemented beyond R2B2.

#### **7.2.1.1.13 Encoder (Class)**

The Encoder interface is implemented by classes representing any type of video encoder. The Encoder interface includes both the Codec and the VideoSendingDevice interfaces, which means in addition to providing forwarding of video, it also is used to send video camera control commands and return responses to a camera control process.

#### **7.2.1.1.14 GeoLocatable (Class)**

This interface is implemented by classes that can provide location information to their users, that is, classes representing real physical world objects which can be described as having a specific real-world location, such as a latitude/longitude, and/or a location along a roadway, and or a textual location description.

#### **7.2.1.1.15 Monitor (Class)**

The Monitor interface is implemented by objects which represent a video monitor, e.g., a real, physical "television set" on which a video image can be displayed. This is the most common type of VideoSink (the other being a SWMonitor, part of a future requirement to stream video directly to user's workstations (or potentially other nearby computers).

#### **7.2.1.1.16 NoVideoAvailableSource (Class)**

The FixedVideoSource interface is implemented by objects which represent a video source other than a video camera, such as the "No Image Available" image generators. This interface could also represent a VCR or any other video source that is not a camera. The FixedVideoSource does not include the GeoLocatable interface because the location (e.g. lat/long) of a fixed video source is irrelevant in CHART II processing (unlike for a VideoCamera, for which the location (lat/long) of a camera could someday be used for automatic identification of cameras near traffic events, automatic pointing of cameras to traffic events, etc.)

#### **7.2.1.1.17 NTCIPCamera (Class)**

The NTCIPCamera interface is implemented by objects which support the NTCIP standard for CCTV cameras.

#### **7.2.1.1.18 PresetEnabled (Class)**

The PresetEnabled interface is implemented by VideoCamera-enabled classes which can store and move to presets. The savePreset() method saves the current camera position as the preset position. This interface is expected to be implemented in R2B2.

#### **7.2.1.1.19 SharedResource (Class)**

The SharedResource interface is implemented by any object that must always have an operations center responsible for the disposition of the resource while the resource is in use.

#### **7.2.1.1.20 SwitchInputPort (Class)**

This is the interface for a switch input port. A switch input port is a type of switch port and is also a type of VideoSendingDevice, meaning it can send a video signal on behalf of the VideoProvider attached to it to any one or more VideoReceivingDevices (and corresponding VideoCollectors).

#### **7.2.1.1.21 SwitchOutputPort (Class)**

This is the interface for a switch output port. A switch output port is a type of switch port and is also a type of VideoReceivingDevice (meaning it receives a video signal on behalf of the VideoCollector attached to it). As VideoReceivingDevice, a SwitchOutputPort is capable of being connected to any VideoSendingDevice.

#### **7.2.1.1.22 SwitchPort (Class)**

The is a generic SwitchPort interface. It is a CommEnabled interface, meaning a SwitchPort can be online or offline. (A SwitchPort cannot be in maintenance mode).

#### **7.2.1.1.23 SWMonitor (Class)**

The SWMonitor interface is implemented by objects which represent a software monitor capable of receiving and displaying video (i.e., a streaming video MPEG software decoder



running on a PC). This interface supports a future requirement to display video directly to user's workstations.

#### **7.2.1.1.24 TransferableSharedResource (Class)**

The TransferrableSharedResource interface is implemented by any object that must always have an operations center responsible for the disposition of the resource while the resource is in use but may also be allowed to transfer control of that resource to another operations center.

#### **7.2.1.1.25 UniquelyIdentifiable (Class)**

The UniquelyIdentifiable interface is implemented by classes whose instances have a unique identifier that is guaranteed not to match the identifier of any other uniquely identifiable objects in the system. It provides access to the unique ID, and the name (which does not have to be unique).

#### **7.2.1.1.26 ViconSVFTCamera (Class)**

The ViconSVFTCamera interface is implemented by a class representing the Vicon Surveyor VFT model video camera. (As there are no other Vicon brand cameras used within CHART II, there is no base ViconCamera interface representing all Vicon-brand cameras. For one thing, there would be no known basis for allocating methods to the base interface and the VFT interface.)

#### **7.2.1.1.27 ViconSVFTPgmCmd (Class)**

The ViconSVFTPgmCmd enumeration defines the values that can be used in the programCommand() method of the ViconSVFTCamera interface.

#### **7.2.1.1.28 VideoCamera (Class)**

The VideoCamera interface is implemented by objects representing video cameras within the CHART II system. Classes implementing this interface (and nothing below this interface would be fixed (non-controllable) video cameras. The VideoCamera interface includes the GeoLocatable interface, to someday allow for advanced features such as automatic identification of cameras near traffic events, automatic pointing of cameras to traffic events, etc.

#### **7.2.1.1.29 VideoCollector (Class)**

The VideoCollector interface is a generic abstract interface including VideoSink objects (e.g. video monitors) and BridgeCircuit objects. Both VideoSink and BridgeCircuit objects collect video from a VideoProvider, but only VideoSink objects are true destination endpoints for video feeds which a typical user would have direct interaction with. BridgeCircuit VideoCollector objects are merely an intermediate step in a VideoRoute which eventually provides video ultimately to the VideoSink object(s) at the end of the route(s).

#### **7.2.1.1.30 VideoFabric (Class)**

The VideoFabric is implemented by a class which represents a "video fabric", that is a collection of VideoTransmissionDevice objects on a common "fabric" across which video can be created directly. This includes any collection of switch input ports and switch output ports on a single video switch. (Note that a collection of encoder and decoder types of VideoTransmissionDevice objects represents a different video fabric, across which video can be routed directly. The IP encoder/decoder fabric therefore is different from other fabrics in that it has no associated video switch.

#### **7.2.1.1.31 VideoFabricConfig (Class)**

The VideoFabricConfig structure is used to store and transmit configuration information about a VideoFabric object.

#### **7.2.1.1.32 VideoProvider (Class)**

The VideoProvider interface is a generic abstract interface including VideoSource objects (e.g. video cameras) and BridgeCircuit objects. Both VideoSource and BridgeCircuit objects provide video to a VideoCollector, but only VideoSource objects are true origins of video which a typical user would have direct interaction with. BridgeCircuit VideoProvider objects merely pass on video provided from elsewhere in a VideoRoute.

#### **7.2.1.1.33 VideoReceivingDevice (Class)**

The VideoReceivingDevice interface is implemented by objects which can be used to receive video from a corresponding VideoSendingDevice. A VideoReceivingDevice may be an MPEG decoder or may be an output port on a video switch.

#### **7.2.1.1.34 VideoRoute (Class)**

This interface defines a route through CHART II video distribution system. A given implementation of a VideoRoute may or may not be actively in use at any given time. Routes are defined by the combinations of all bridge circuits between all pairs of switch fabrics within the CHART II video distribution system. Routes cannot be added or deleted or enabled or disabled by users explicitly: the routes and their status are defined implicitly by the configuration and status of bridge circuits defined in the system at any given time.

#### **7.2.1.1.35 VideoRouteManager (Class)**

The VideoRouteManager interface is implemented by a class which provides video routing capabilities within CHART II. This router does not need to be used (in fact, cannot be used) when the VideoSource and VideoSink are on the same switch fabric -- it is used only to make video routes across switch fabrics. The implementation will use a set of rules to arbitrate among requested video displays when a set of bridge circuits between one or more pairs of switch fabrics is fully utilized. Based on the override rules implemented, a new incoming routing request may or may not be able to be fulfilled depending upon priority, routing guarantees, number of images viewed, ongoing camera control sessions, etc. If an override can be granted, the overridden route(s) will be dropped in favor of the new route.

#### **7.2.1.1.36 VideoSendingDevice (Class)**

The VideoSendingDevice interface is implemented by objects which can be used to send video to a corresponding VideoReceivingDevice. A VideoSendingDevice may be an MPEG encoder or may be an input port on a video switch.

#### **7.2.1.1.37 VideoSink (Class)**

The VideoSink interface is implemented by objects which serve as final endpoints for video signals, such as video monitors and streaming video receivers directly on user workstations. Within the user interface, the VideoSink interface represents all objects on which a video source can be placed for viewing by users.

#### **7.2.1.1.38 VideoSinkInfo (Class)**

VideoSinkInfo represent information about a VideoSink that is used by a VideoTour.

#### **7.2.1.1.39 VideoSource (Class)**

The VideoSource interface is implemented by objects which originate video signals, such as video cameras and image generators. Within the user interface, the VideoSource interface represents all video sources which can be put on monitors (i.e., VideoSink objects).

The VideoSource interface includes the SharedResource interface. A VideoSource is controlled by an Operations Center if the VideoSource is in maintenance mode, or if the VideoSource is a camera which has an active control session up.

#### **7.2.1.1.40 VideoSwitch (Class)**

The V1500Switch interface is implemented by a class representing any V1500 Video Switch in the CHART system. This interface provides access to configuration and status information for the switch, and provides connect and disconnect functions for making and breaking video connections.

#### **7.2.1.1.41 VideoSwitchConfig (Class)**

This represents the configuration information for a V1500 switch.

#### **7.2.1.1.42 VideoTour (Class)**

The Tour interface is implemented by a class which maintains the configuration and status of a single tour defined within the CHART II system.

#### **7.2.1.1.43 VideoTourConfig (Class)**

The TourConfig structure is used to hold and transmit configuration information about a given camera tour.

#### **7.2.1.1.44 VideoTourEntry (Class)**

The TourEntry structure is used to hold and transmit configuration information about a single entry in a camera tour.

#### **7.2.1.1.45 VideoTourFactory (Class)**

The TourManager interface is implemented by a class which tracks tours defined in the CHART II video system. It tracks the existence and configuration of tours and also tracks the status of all tours, whether they are active or not.

#### **7.2.1.1.46 VideoTourInfo (Class)**

A structure of related information about a single VideoTour.

#### **7.2.1.1.47 VideoTourState (Class)**

The VideoTourState enumeration defines the values that can be used to indicate the status of a VideoTour.

#### **7.2.1.1.48 VideoTourStatus (Class)**

The TourStatus structure is used to hold and transmit status information about a given camera tour (e.g., what VideoSink objects the Tour is currently running on).

#### **7.2.1.1.49 VideoTransmissionDevice (Class)**

The VideoTransmissionDevice interface is implemented by objects representing devices which can be used for sending and receiving video. This interface provides CHART-standard methods for accessing status and configuration information. Specific interfaces supporting sending and receiving inherit from this abstract base interface.

### **7.2.1.2 VideoHighLevel-VideoSource (Class Diagram)**

This diagram shows the VideoSource side of the VideoHighLevel diagram in more detail, adding Factories, Configuration and Status structures, exceptions, and other supporting interface elements. In general each of the major interface objects, VideoProvider, VideoSource, VideoCamera, and ControllableVideoCamera have a factory and configuration and status structures used to store and transmit configuration and status information to clients and interested server objects.



#### **7.2.1.2.4 CameraNotControlledException (Class)**

This is an exception thrown if an attempt to issue a camera control command is issued when the camera is not currently controlled by the requester. This is most likely to occur immediately after a control override, in cases where the client has not received or processed the override event yet.

#### **7.2.1.2.5 CameraPreset (Class)**

This structure stores information about a preset configured for a camera.

#### **7.2.1.2.6 COHU3955Camera (Class)**

The COHU3955Camera interface is implemented by objects representing COHU model 3055 video cameras. It extends the ControllableVideoCamera interface by adding methods unique to the COHU 3955 cameras (unique within the universe of camera types planned for implementation within CHART II).

#### **7.2.1.2.7 COHU3955CameraStatus (Class)**

The COHUCameraStatus structure is used to hold status information about COHUCamera objects at the COHUCamera level.

#### **7.2.1.2.8 ControllableVideoCamera (Class)**

The ControllableVideoCamera interface is implemented by objects representing controllable video cameras within the CHART II system. The ControllableVideoCamera interface represents a controllable video camera as opposed to an uncontrollable, immovable VideoCamera. Current plans call for classes to represent a COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera, and there are interfaces defined for each of these subtypes of ControllableVideoCamera. The ControllableVideoCamera interface includes all methods common to the two known types of video cameras currently in use by MDSHA, although it is likely to contain a superset of methods which would be implemented by the entire universe of all video cameras which could someday be used. This interface may have to be refined in the event that future brands or models of video cameras might be incorporated under CHART II, but it is an appropriate set of methods for the present day. Current plans call for classes to represent a COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera.

#### **7.2.1.2.9 ControllableVideoCameraConfig (Class)**

The ControllableVideoCameraConfig is used to hold and transmit configuration information about ControllableVideoCamera objects at the ControllableVideoCamera level.

#### **7.2.1.2.10 ControllableVideoCameraStatus (Class)**

The ControllableVideoCameraStatus is used to hold and transmit status information about ControllableVideoCameraStatus objects at the ControllableVideoCamera level.

#### **7.2.1.2.11 ControllingInfo (Class)**

The ControllingInfo structure contains information about the entity controlling (or requesting to control) a VideoCamera.

#### **7.2.1.2.12 ControllingUserInfo (Class)**

The ControllingUserInfo structure contains information about the monitor group and user of the entity controlling (or requesting to control) a VideoCamera.

#### **7.2.1.2.13 DiagonallyMovable (Class)**

The DiagonallyMovable interface is implemented by VideoCamera-enabled classes which can be moved diagonally in addition to standard orthogonal pan and tilt commands. A particular implementation may support 45-degree movements only, in which case the panDir and tiltDir parameters are +/- 1 to indicate direction only, or an implementation may support 360 degrees of motion, in which case, in addition to signs, the relative ratios of the parameters indicate the percent of movement proportionally in the pan/tilt directions. This interface is expected to be implemented beyond R2B2.

#### **7.2.1.2.14 EnMasseSetResult (Class)**

This structure will be used to communicate failures in setting a number of cameras to auto iris, auto focus, or auto color balance. It specifies results for one camera which failed.

#### **7.2.1.2.15 EnMasseSetResultList (Class)**

This structure will be used to communicate failures in setting a number of cameras to auto iris, auto focus, or auto color balance. It specifies results for all cameras which failed. (Cameras which succeeded are not included in this list.)

#### **7.2.1.2.16 VideoControlFlashConfig (Class)**

This structure stores configuration information about a flash streaming server configuration that is displaying a camera's image.

#### **7.2.1.2.17 MonitorDisplayInfo (Class)**

This structure holds details about each monitor on which the VideoProvider is currently being displayed.

#### **7.2.1.2.18 NTCIPCamera (Class)**

The NTCIPCamera interface is implemented by objects which support the NTCIP standard for CCTV cameras.

#### **7.2.1.2.19 PresetEnabled (Class)**

The PresetEnabled interface is implemented by VideoCamera-enabled classes which can store and move to presets. The savePreset() method saves the current camera position as

the preset position. This interface is expected to be implemented in R2B2.

#### **7.2.1.2.20 PresetUndefinedException (Class)**

This exception is thrown when an attempt is made to move to an undefined preset.

#### **7.2.1.2.21 SharedResource (Class)**

The SharedResource interface is implemented by any object that must always have an operations center responsible for the disposition of the resource while the resource is in use.

#### **7.2.1.2.22 SharedResourceManager (Class)**

The SharedResourceManager interface is implemented by classes that manage shared resources. Implementing classes must be able to provide a list of all shared resources under their management. Implementing classes must also be able to tell others if there are any resources under its management that are controlled by a given operations center.

#### **7.2.1.2.23 TransferableSharedResource (Class)**

The TransferrableSharedResource interface is implemented by any object that must always have an operations center responsible for the disposition of the resource while the resource is in use but may also be allowed to transfer control of that resource to another operations center.

#### **7.2.1.2.24 ViconSVFTCamera (Class)**

The ViconSVFTCamera interface is implemented by a class representing the Vicon Surveyor VFT model video camera. (As there are no other Vicon brand cameras used within CHART II, there is no base ViconCamera interface representing all Vicon-brand cameras. For one thing, there would be no known basis for allocating methods to the base interface and the VFT interface.)

#### **7.2.1.2.25 ViconSVFTPgmCmd (Class)**

The ViconSVFTPgmCmd enumeration defines the values that can be used in the programCommand() method of the ViconSVFTCamera interface.

#### **7.2.1.2.26 VideoCamera (Class)**

The VideoCamera interface is implemented by objects representing video cameras within the CHART II system. Classes implementing this interface (and nothing below this interface would be fixed (non-controllable) video cameras. The VideoCamera interface includes the GeoLocatable interface, to someday allow for advanced features such as automatic identification of cameras near traffic events, automatic pointing of cameras to traffic events, etc.



#### **7.2.1.2.27 VideoCameraConfig (Class)**

The VideoCameraConfig structure is used to hold configuration information about VideoCamera objects at the VideoCamera level. Further details about lower-level VideoCamera subclasses are provided by subclasses of VideoCameraConfig. Contains a detailed location information.

#### **7.2.1.2.28 VideoCameraFactory (Class)**

The GenericVideoCameraFactory interface is implemented by factory classes responsible for creating, maintaining, and controlling a collection of GenericVideoCamera objects.

#### **7.2.1.2.29 VideoCameraStatus (Class)**

The VideoCameraStatus structure is used to hold status information about VideoCamera objects at the VideoCamera level. Further details about lower-level VideoCamera subclasses are provided by subclasses of VideoCameraStatus.

#### **7.2.1.2.30 VideoDisplayRevokedOrg (Class)**

This structure is used to store information about an organization for which display of the associated camera has been revoked.

#### **7.2.1.2.31 VideoProvider (Class)**

The VideoProvider interface is a generic abstract interface including VideoSource objects (e.g. video cameras) and BridgeCircuit objects. Both VideoSource and BridgeCircuit objects provide video to a VideoCollector, but only VideoSource objects are true origins of video which a typical user would have direct interaction with. BridgeCircuit VideoProvider objects merely pass on video provided from elsewhere in a VideoRoute.

#### **7.2.1.2.32 VideoProviderConfig (Class)**

The VideoProviderConfig structure is used to hold configuration information about VideoProvider objects at the VideoProvider level. The object contains the provider name, provider type, video component type, owning org, network connection site, sending device IDs and sending device configurations. Further details about lower-level VideoProvider subclasses are provided by subclasses of VideoProviderConfig.

#### **7.2.1.2.33 VideoProviderFactory (Class)**

The VideoProviderFactory interface is implemented by factory classes responsible for creating and maintaining a collection of VideoProvider objects.

#### **7.2.1.2.34 VideoProviderStatus (Class)**

The VideoProviderStatus structure is used to hold status information about VideoProvider objects at the VideoProvider level. Further details about lower-level VideoProvider subclasses are provided by subclasses of VideoProviderStatus.

#### **7.2.1.2.35 VideoSource (Class)**

The VideoSource interface is implemented by objects which originate video signals, such as video cameras and image generators. Within the user interface, the VideoSource interface represents all video sources which can be put on monitors (i.e., VideoSink objects).

The VideoSource interface includes the SharedResource interface. A VideoSource is controlled by an Operations Center if the VideoSource is in maintenance mode, or if the VideoSource is a camera which has an active control session up.

#### **7.2.1.2.36 VideoSourceConfig (Class)**

The VideoSourceConfig structure is used to hold configuration information about VideoSource objects at the VideoSource level. It contains a video provider configuration, a boolean NVA indicator, and a video control flash configuration. Further details about lower-level VideoSource subclasses are provided by subclasses of VideoSourceConfig.

#### **7.2.1.2.37 VideoSourceFactory (Class)**

The VideoSourceFactory interface is implemented by factory classes responsible for creating, maintaining, and controlling a collection of VideoSource objects.

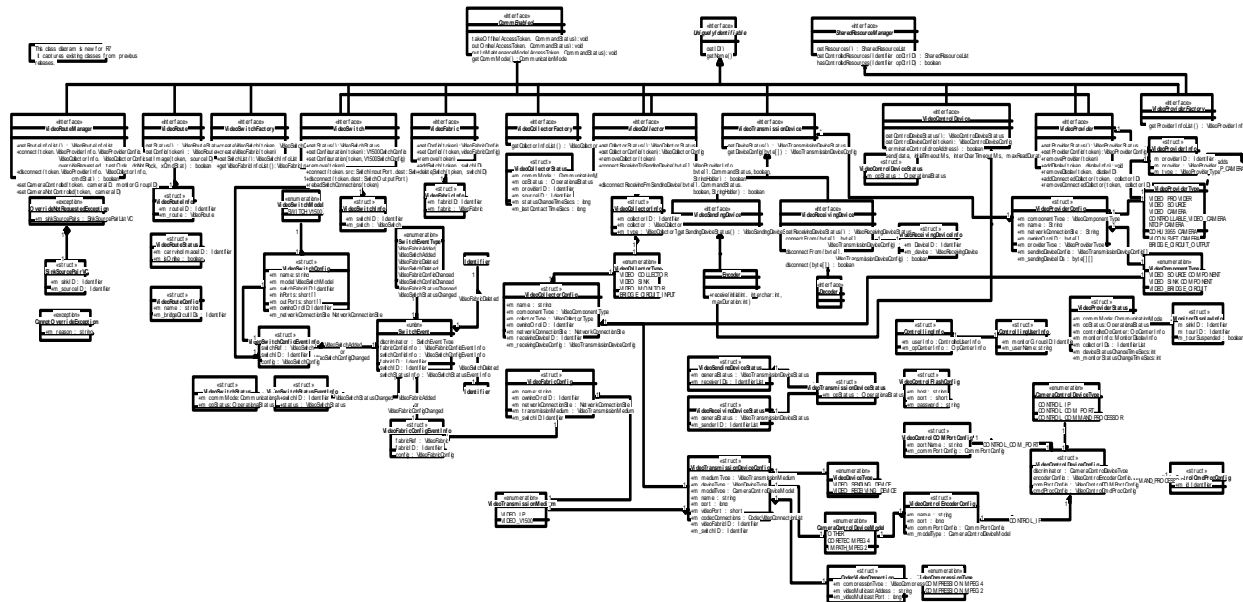
#### **7.2.1.2.38 VideoSourceStatus (Class)**

The VideoSourceStatus structure is used to hold status information about VideoSource objects at the VideoSource level. Further details about lower-level VideoSource subclasses are provided by subclasses of VideoSourceStatus.

#### **7.2.1.2.39 VideoSourceType (Class)**

This enumeration identifies the various types of cameras which can exist in the system. The fixed type is for all non-controllable cameras.

### 7.2.1.3 VideoControl (Class Diagram)



### Figure 7-6 VideoControl (Class Diagram)

#### 7.2.1.3.1 CameraControlDeviceModel (Class)

This enum lists the models of camera control devices used.

#### 7.2.1.3.2 CameraControlDeviceType (Class)

This enum lists the types of control devices which can be used to control video cameras. This enum is used as the discriminator in the VideoControlDeviceConfig union.

#### 7.2.1.3.3 CannotOverrideException (Class)

This exception is thrown when attempt is made to display an image with requires a route, but the route could not be made due to higher priority routes already existing.

#### 7.2.1.3.4 CodecVideoConnection (Class)

This structure defines configuration data pertaining to one specific CODEC transmission stream.

#### 7.2.1.3.5 CommEnabled (Class)

The CommEnabled interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically

apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by TrafficEvents within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by TrafficEvents), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

#### **7.2.1.3.6 ControllingInfo (Class)**

The ControllingInfo structure contains information about the entity controlling (or requesting to control) a VideoCamera.

#### **7.2.1.3.7 ControllingUserInfo (Class)**

The ControllingUserInfo structure contains information about the monitor group and user of the entity controlling (or requesting to control) a VideoCamera.

#### **7.2.1.3.8 Decoder (Class)**

This interface describes the Decoder interface. The decoder includes the VideoReceivingDevice interface.

#### **7.2.1.3.9 Encoder (Class)**

The Encoder interface is implemented by classes representing any type of video encoder. The Encoder interface includes both the Codec and the VideoSendingDevice interfaces, which means in addition to providing forwarding of video, it also is used to send video camera control commands and return responses to a camera control process.

#### **7.2.1.3.10 Identifier (Class)**

Wrapper class for a CHART2 identifier byte sequence. This class will be used to add identifiable objects to hash tables and perform subsequent lookup operations.

#### **7.2.1.3.11 MonitorDisplayInfo (Class)**

This object is used to identify a video sink currently displaying video from a video provider. A List of these structures is stored in a video provider's status.

#### **7.2.1.3.12 OverrideNotRequestedException (Class)**

This exception is thrown when attempt is made to display an image which requires a route, but the route could not be made due to all routes already in use. The implication is that if override had been requested, the route would be likely to be created (which would override another route, or routes). Information is provided about what monitor(s) would be likely to be overridden, along with what source each monitor is viewing.

#### **7.2.1.3.13 SharedResourceManager (Class)**

The SharedResourceManager interface is implemented by classes that manage shared resources. Implementing classes must be able to provide a list of all shared resources under their management. Implementing classes must also be able to tell others if there are any resources under its management that are controlled by a given operations center. The shared resource manager is also responsible for periodically monitoring its shared resources to detect if the operations center controlling a resource doesn't have at least one user logged into the system. When this condition is detected, the shared resource manager must push an event on the ResourceManagement event channel to notify others of this condition.

#### **7.2.1.3.14 SinkSourcePairVC (Class)**

This structure contains a VideoSink ID and VideoSource ID pair.

#### **7.2.1.3.15 SwitchEvent (Class)**

This structure stores configuration information used to find and use the video control device used to send/receive camera control commands/responses to/from a camera.

#### **7.2.1.3.16 SwitchEventType (Class)**

This enum lists the events related to switch control (switches and fabrics) that are pushed on a switch event channel through the CORBA event service. The data pushed with these events is defined in the SwitchEvent union.

#### **7.2.1.3.17 UniquelyIdentifiable (Class)**

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

#### **7.2.1.3.18 VideoCollector (Class)**

The VideoCollector interface is a generic abstract interface including VideoSink objects (e.g. video monitors) and BridgeCircuit objects. Both VideoSink and BridgeCircuit objects collect video from a VideoProvider, but only VideoSink objects are true destination endpoints for video feeds which a typical user would have direct interaction with. BridgeCircuit VideoCollector objects are merely an intermediate step in a VideoRoute which eventually provides video ultimately to the VideoSink object(s) at the end of the route(s).

#### **7.2.1.3.19 VideoCollectorConfig (Class)**

This structure defines configuration data common to all video collectors.

#### **7.2.1.3.20 VideoCollectorFactory (Class)**

This interface defines an object that is used to manage video collector objects in the system. There is no create operation because VideoCollector is an abstract interface.

#### **7.2.1.3.21 VideoCollectorInfo (Class)**

A structure of related information about a single VideoCollector.

#### **7.2.1.3.22 VideoCollectorStatus (Class)**

This structure defines the data included in the status of a video collector.

#### **7.2.1.3.23 VideoCollectorType (Class)**

This enum lists the different types of VideoCollector in the system.

#### **7.2.1.3.24 VideoComponentType (Class)**

This enum lists the video component types supported by the software.

#### **7.2.1.3.25 VideoCompressionType (Class)**

This enum lists the models of camera control devices used.

#### **7.2.1.3.26 VideoControlCmdProcConfig (Class)**

This structure stores configuration information about a command processor based video control device used to transmit camera control commands/responses to/from the camera. This is the structure in a VideoControlDeviceConfig if its CameraControlDeviceType discriminator is CONTROL\_COMMAND\_PROCESSOR.

#### **7.2.1.3.27 VideoControlCOMPortConfig (Class)**

This structure stores configuration information about a COM port based video control device used to transmit camera control commands/responses to/from the camera. This structure is in a VideoControlDeviceConfig if its CameraControlDeviceType discriminator is CONTROL\_COM\_PORT.

#### **7.2.1.3.28 VideoControlDevice (Class)**

This interface is used to represent a video control device in the field. A video control device is used to communicate camera control commands to a camera, and return responses to the requester.

#### **7.2.1.3.29 VideoControlDeviceConfig (Class)**

This structure stores configuration information used to find and use the video control device used to send/receive camera control commands/responses to/from a camera.

#### **7.2.1.3.30 VideoControlDeviceStatus (Class)**

This structure defines status data common to a video control device, i.e. ,a device used to send/receive camera control commands/responses to/from the camera.

#### **7.2.1.3.31 VideoControlEncoderConfig (Class)**

This structure stores configuration information about an IP encoder based video control device used to transmit camera control commands/responses to/from the camera. This structure is in a VideoControlDeviceConfig if its CameraControlDeviceType discriminator is CONTROL\_IP.

#### **7.2.1.3.32 VideoControlFlashConfig (Class)**

This structure stores configuration information about a flash streaming server configuration that is displaying a camera's image.

#### **7.2.1.3.33 VideoDeviceType (Class)**

This enum lists the types video transmission devices that a video transmission device can be.

#### **7.2.1.3.34 VideoFabric (Class)**

The VideoFabric interface is implemented by a class representing any Video Fabric in the CHART system. This interface provides access to configuration for the video fabric.

#### **7.2.1.3.35 VideoFabricConfig (Class)**

This class contains the configuration information for a given VideoFabric.

#### **7.2.1.3.36 VideoFabricConfigEventInfo (Class)**

This struct is used for passing event data related to a video fabric configuration when a video fabric is added to the system or undergoes a configuration change.

#### **7.2.1.3.37 VideoFabricInfo (Class)**

A structure of related information about a single VideoFabric.

#### **7.2.1.3.38 VideoProvider (Class)**

The VideoProvider interface is a generic abstract interface including VideoSource objects (e.g. video cameras) and BridgeCircuit objects. Both VideoSource and BridgeCircuit objects provide video to a VideoCollector, but only VideoSource objects are true origins of video which a typical user would have direct interaction with. BridgeCircuit VideoProvider objects merely pass on video provided from elsewhere in a VideoRoute.

#### **7.2.1.3.39 VideoProviderConfig (Class)**

This structure defines configuration data common to all video sources.

#### **7.2.1.3.40 VideoProviderFactory (Class)**

This interface defines an object that is used to manage video provider objects in the system. There is no create operation because VideoProvider is an abstract interface.

#### **7.2.1.3.41 VideoProviderInfo (Class)**

A structure of related information about a single VideoProvider.

#### **7.2.1.3.42 VideoProviderStatus (Class)**

The VideoProviderStatus structure is used to hold and transmit status information about VideoProvider objects at the VideoProvider level. Further details about lower-level VideoProvider subclasses are provided by subclasses of VideoProviderStatus.

#### **7.2.1.3.43 VideoProviderType (Class)**

This enum lists the different types of VideoProvider in the system.

#### **7.2.1.3.44 VideoReceivingDevice (Class)**

The VideoReceivingDevice interface is used to represent a video receiving device in the field. These devices are used to actually connect a video provider to a video collector. The system contains an instance of this interface for each video receiving device.

#### **7.2.1.3.45 VideoReceivingDeviceInfo (Class)**

A tuple of related information about a single VideoReceivingDevice.

#### **7.2.1.3.46 VideoReceivingDeviceStatus (Class)**

This structure defines status data for video receiving devices.

#### **7.2.1.3.47 VideoRoute (Class)**

This interface defines the operations for a video route.

#### **7.2.1.3.48 VideoRouteConfig (Class)**

This structure defines configuration data for a VideoRoute.

#### **7.2.1.3.49 VideoRouteInfo (Class)**

A structure of related information about a single VideoRoute.

#### **7.2.1.3.50 VideoRouteManager (Class)**

The VideoRouteManager interface is implemented by a class which provides video routing capabilities within CHART II. This router does not need to be used (in fact, cannot be used) when the VideoSource and VideoSink are on the same switch fabric -- it is used only to make video routes across switch fabrics.

#### **7.2.1.3.51 VideoRouteStatus (Class)**

This structure defines status data for a video route.



#### **7.2.1.3.52 VideoSendingDevice (Class)**

The VideoSendingDevice interface is implemented by objects which can be used to send video to a corresponding VideoReceivingDevice. A VideoSendingDevice may be an MPEG encoder or may be an input port on a video switch.

#### **7.2.1.3.53 VideoSendingDeviceStatus (Class)**

The VideoSendingDeviceStatus structure is used to store generic status information common to all types of VideoSendingDevice objects. Subclasses will provide additional information specific to the type of object/interface referenced at that level of the VideoTransmissionDevice inheritance tree at that point.

#### **7.2.1.3.54 VideoSwitch (Class)**

The V1500Switch interface is implemented by a class representing any V1500 Video Switch in the CHART system. This interface provides access to configuration and status information for the switch, and provides connect and disconnect functions for making and breaking video connections.

#### **7.2.1.3.55 VideoSwitchConfig (Class)**

This represents the configuration information for a V1500 switch (R2B2).

#### **7.2.1.3.56 VideoSwitchConfigEventInfo (Class)**

This struct is used for passing event data related to a video switch configuration when a video switch is added to the system or undergoes a configuration change.

#### **7.2.1.3.57 VideoSwitchFactory (Class)**

The VideoSwitchFactory interface is used to create and manage VideoSwitch objects and SwitchFabric objects in the system.

#### **7.2.1.3.58 VideoSwitchInfo (Class)**

A structure of related information about a single VideoSwitch.

#### **7.2.1.3.59 VideoSwitchModel (Class)**

This enum lists the models video switches a VideoSwitch can be.

#### **7.2.1.3.60 VideoSwitchStatus (Class)**

This represents the status information for a V1500 switch (R2B2).

#### **7.2.1.3.61 VideoSwitchStatusEventInfo (Class)**

This struct is used for passing event data related to a video switch status when a video switch undergoes a status change.

#### **7.2.1.3.62 VideoTransmissionDevice (Class)**

The VideoTransmissionDevice interface is used to represent a video transmission device in the field (either a video sending device or a video receiving device). These devices are used to actually connect a video provider to a video collector. The system contains an instance of this interface for each video transmission device.

#### **7.2.1.3.63 VideoTransmissionDeviceConfig (Class)**

This structure defines configuration data common to all video transmission devices.

#### **7.2.1.3.64 VideoTransmissionDeviceStatus (Class)**

This structure defines status data common to all video transmission devices.

#### **7.2.1.3.65 VideoTransmissionMedium (Class)**

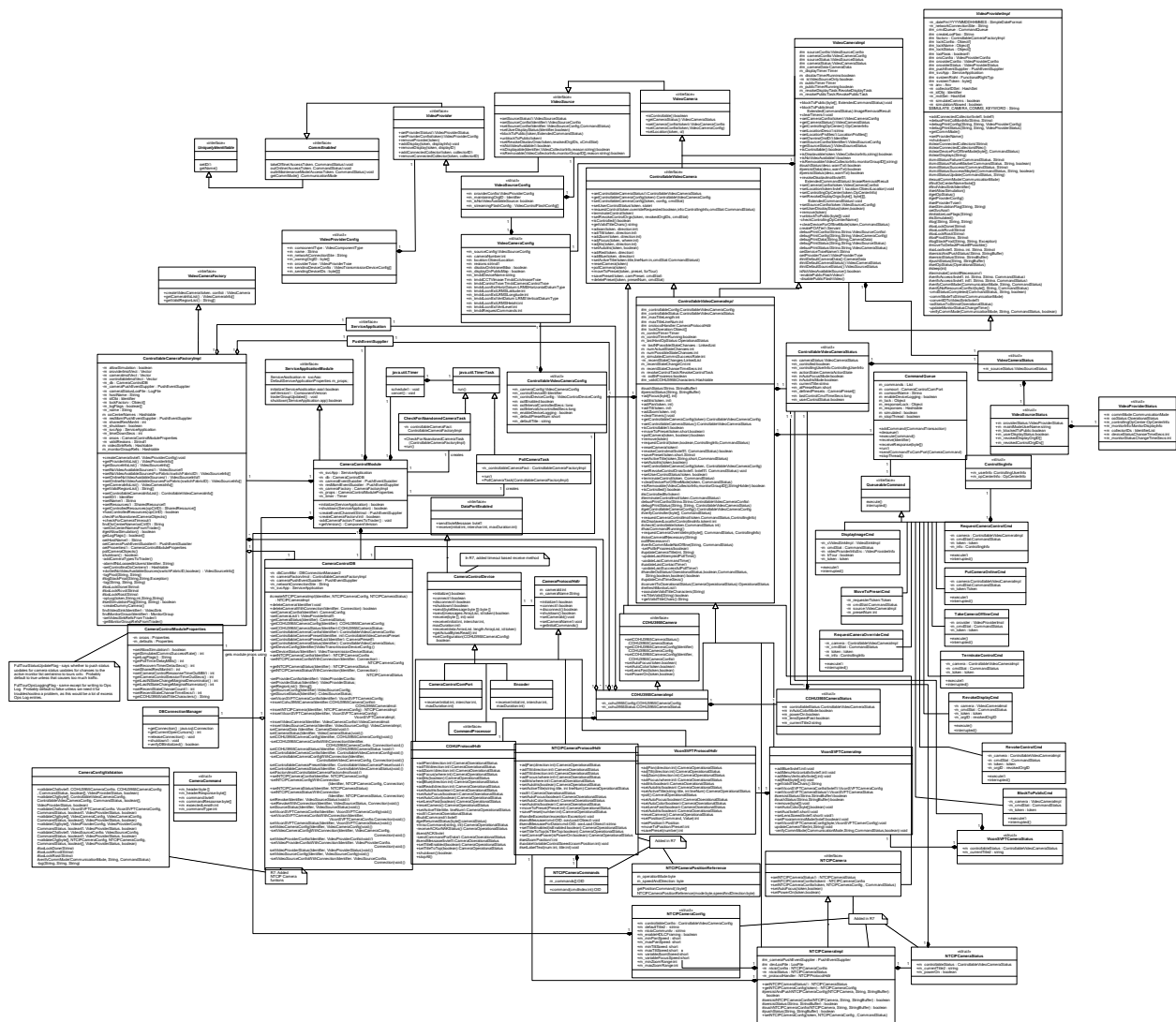
This enum lists the video transmission media supported by the software.

## **7.3 Camera Control Module**

### **7.3.1 Class Diagrams**

#### **7.3.1.1 CameraControlModule (Class Diagram)**

This diagram shows the classes with comprise the CameraControlModule. The CameraControlModule is an installable module that serves the camera-type objects and factories to the rest of the CHART II system. This diagram shows how the implementation of these CORBA interfaces relays on other supporting classes to perform their functions. The CameraControlModule is responsible for serving all VideoSource objects including controllable cameras, fixed cameras, No Video Available sources, and potentially any other image generators, etc. The COHU3955CameraImpl, viconSVFTCameraImpl, and NTCIPCcameraImpl are the primary classes operating in this module. These objects provide all access to the camera status and configuration. The CameraControlModule also includes factory implementations responsible for providing lists of cameras and other such objects to interested clients.



**Figure 7-7. CameraControlModule (Class Diagram)**

#### 7.3.1.1.1 BlockToPublicCmd (Class)

This class represents the information needed to create a block camera to public command to be added on the CommandQueue.

#### 7.3.1.1.2 CameraCommand (Class)

CameraCommand contains information about the commands sent to, and responses received from, the camera.

#### 7.3.1.1.3 CameraConfigValidation (Class)

This class validates camera configuration data for any type of camera (Video Source, (Fixed) Video Camera, COHU3955, SVFT, and NTCIP).

#### **7.3.1.1.4 CameraControlComPort (Class)**

The CameraControlComPort interface is implemented by a class representing a COM port with direct connection to the control port of a video camera. It is used to send video camera control commands and return responses to a camera control process.

#### **7.3.1.1.5 CameraControlDB (Class)**

The CameraControlDB class provides an interface between the Camera service and the database used to persist and depersist the Camera objects and their configuration and status in the database. It contains a collection of methods that perform database operations on tables pertinent to Camera Control. The class is constructed with a DBConnectionManager object, which manages database connections. Methods exist to insert and delete Camera objects from the database, and to get and set their configuration and status information.

#### **7.3.1.1.6 CameraControlDevice (Class)**

The CameraControlDevice interface is implemented by classes which provide communications for access to control functions for a video camera. This includes encoders, command processors, and direct COM ports.

#### **7.3.1.1.7 CameraControlModule (Class)**

The CameraControlModule class is the service module for the Camera devices and a Camera factory. It implements the ServiceApplicationModule interface. It creates and serves a single CameraFactoryImpl object, which in turn serves zero or more CameraImpl objects. It also creates CameraControlDB, CameraControlModuleProperties, and PushEventSupplier objects.

#### **7.3.1.1.8 CameraControlModuleProperties (Class)**

The CameraControlModuleProperties class is used to provide access to properties used by the Camera Control Module. This class wraps properties that are passed to it upon construction. It adds its own defaults and provides methods to extract properties specific to the Camera Control Module.

#### **7.3.1.1.9 CameraProtocolHdlr (Class)**

CameraProtocolHdlr classes provide implementations for all the camera commands. Each CameraImpl class will have a CameraProtocolHdlr instantiated when initialized. When a camera control command is sent to the CameraImpl, CameraProtocolHdlr will be called to translate the command to byte messages which the camera understands. Then those messages are sent by the CameraControlDevice to the camera. CameraProtocolHdlr is capable of using different CameraControlDevice which is created during the initialization.

#### **7.3.1.1.10 CheckForAbandonedCameraTask (Class)**

The CheckForAbandonedCameraTask is a timer task. When the timer fires, it checks to see if a camera control session has exceeded the timeout, or whether a camera is controlled by

an Operations center with no one logged in.

#### **7.3.1.1.11 COHU3955Camera (Class)**

The COHUCamera interface is implemented by objects representing COHU-brand video cameras. The COHUCamera interface is extended by the COHUMPCCamera and COHU3955Camera interfaces. The COHUCamera interface includes all methods which are common to the two COHU cameras used by CHART II, the COHU MPC camera and the COHU 3955 camera. (Note that this interface may well contain a superset of methods which would be implemented by the entire line of all models of COHU video cameras).

#### **7.3.1.1.12 COHU3955CameraImpl (Class)**

This class implements the COHU3955Camera interface, and inherits from the ControllableCameraImpl class. The COHU3955CameraImpl implements methods of COHU3955Camera, extending the controllable camera to include 3955-specific operations. This class will contain a configuration and status object as necessary to convey 3955-specific configuration and status information.

#### **7.3.1.1.13 COHU3955CameraStatus (Class)**

The CameraStatus class is an abstract value-type class which provides status information for a Camera. This status information is relatively dynamic: things like the communication mode, operational status, operation center information, status change time.

#### **7.3.1.1.14 COHUProtocolHdlr (Class)**

COHUProtocolHdlr is the base class for all COHU cameras. At present, this class contains implementations for common functions for COHU MPC and COHU 3955 cameras

#### **7.3.1.1.15 CommandProcessor (Class)**

The CommandProcessor interface is implemented by a class representing a command processor control port with direct connection to the control port of several video cameras. It is used to send video camera control commands and return responses to a camera control process.

#### **7.3.1.1.16 CommandQueue (Class)**

The CommandQueue class provides a queue for QueueableCommand objects. The CommandQueue has a thread that it uses to process each QueueableCommand in a first in first out order. As each command object is pulled off the queue by the CommandQueue's thread, the command object's execute method is called, at which time the command performs its intended task.

#### **7.3.1.1.17 CommEnabled (Class)**

The CommEnabled interface is implemented by objects that can be taken offline, put online, or put in maintenance mode through a standard interface. These states typically

apply only to field devices. When a device is taken offline, it is no longer available for use through the system and automated polling (if any) is halted. When put online, a device is again available for use by TrafficEvents within the system and automated polling is enabled (if applicable). When put in maintenance mode a device is offline (i.e., cannot be used by TrafficEvents), and maintenance commands appropriate for the particular type of device are allowed to help in troubleshooting.

#### **7.3.1.1.18 ControllableCameraFactoryImpl (Class)**

The CameraFactoryImpl class provides an implementation of the CameraFactory interface (and its CameraFactory and SharedResourceManager interfaces) as specified in the IDL. The CameraFactoryImpl maintains a list of CameraImpl objects and is responsible for publishing Camera objects in the Trader on startup and as new camera objects are created. Whenever a Camera is created or removed, that information is persisted to the database. This class is also responsible for performing the checks requested by the timer tasks: to poll the Camera devices, to look for Camera devices with timeout exceeded, to look for Camera devices with no one logged in at the controlling operations center, and to initiate recovery processing as needed

#### **7.3.1.1.19 ControllableVideoCamera (Class)**

The ControllableVideoCamera interface is implemented by objects representing controllable video cameras within the CHART II system. The ControllableVideoCamera interface represents a controllable video camera as opposed to the uncontrollable, immovable VideoCamera. Current plans call for classes to represent a COHU MPC camera, COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera, and there are interfaces defined for each of these subtypes of ControllableVideoCamera. The ControllableVideoCamera interface includes all methods common to the three known types of video cameras currently in use by MDSHA, although it is likely to contain a superset of methods which would be implemented by the entire universe of all video cameras which could someday be used. This interface may have to be refined in the event that future brands or models of video cameras might be incorporated under CHART II, but it is an appropriate set of methods for the present day. Current plans call for classes to represent a COHU MPC camera, COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera.

#### **7.3.1.1.20 ControllableVideoCameraConfig (Class)**

The ControllableVideoCameraConfig is used to hold and transmit configuration information about ControllableVideoCamera objects at the ControllableVideoCamera level.

#### **7.3.1.1.21 ControllableVideoCameraImpl (Class)**

The ControllableCameraImpl class provides an implementation of the ControllableVideoCamera interface and is derived from the CameraImpl class implementing the VideoCamera interface.

This class contains a CommandQueue object that is used to sequentially execute long

running operations related to camera control in a thread separate from the CORBA request threads, thus allowing quick initial responses.

Also contained in this class are ControllableVideoCameraConfig and ControllableVideoCameraStatus objects (used to store the configuration and status of the camera), and a VideoCameraData object (used to store internal status information which is persisted but not pushed out to clients).

The ControllableCameraImpl contains \*Impl methods that map to methods specified in the IDL, including requests to request control of the camera, terminate control of the camera, override control of the camera, and to send pan/tilt/zoom (PTZ) commands to the camera. Some of these requests are long running, so each request is stored in a specific subclass of QueueableCommand and added to the CommandQueue. The queueable command objects simply call the appropriate ControllableCameraImpl method as the command is executed by the CommandQueue in its thread of execution. PTZ commands are not considered long running and are not placed on the command queue.

The ControllableCameraImpl also contains methods called by the CameraFactory to support the timer tasks of the Camera Service: to poll the Camera, to look for Camera devices with communications timeout exceeded.

#### **7.3.1.1.22 ControllableVideoCameraStatus (Class)**

The ControllableVideoCameraStatus is used to hold and transmit status information about ControllableVideoCameraStatus objects at the ControllableVideoCamera level.

#### **7.3.1.1.23 ControllingInfo (Class)**

The ControllingInfo structure contains information about the entity controlling (or requesting to control) a VideoCamera.

#### **7.3.1.1.24 DataPortEnabled (Class)**

This interface is implemented by device specific communications classes. This interface provides an extra layer to remove dependencies on device specific packages.

#### **7.3.1.1.25 DBConnectionManager (Class)**

This class implements a database connection manager that manages a pool of database connections. Any CHART II system thread requiring database access gets a database connection from the pool of connections maintained by this manager class. The connections are maintained in two separate lists namely, inUseList and freeList. The inUseList contains connections that have already been assigned to a thread. The freeList contains unassigned connections. This class assumes that an appropriate JDBC driver has been loaded either by using the "jdbc.drivers" system property or by loading it explicitly. The class has a monitor thread that is started by the constructor. This connection monitor thread periodically checks the inUseList to see if there are connections that are owned by dead threads and move such connections to the freeList. The connection monitor thread is started only if a non-zero value is specified for the monitoring time interval in the constructor.

#### **7.3.1.1.26 DisplayImageCmd (Class)**

This class represents the information needed to create a display image command to be added on the CommandQueue.

#### **7.3.1.1.27 Encoder (Class)**

The Encoder interface is implemented by classes representing any type of video encoder. The Encoder interface includes both the Codec and the VideoSendingDevice interfaces, which means in addition to providing forwarding of video, it also is used to send video camera control commands and return responses to a camera control process.

#### **7.3.1.1.28 java.util.Timer (Class)**

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

#### **7.3.1.1.29 java.util.TimerTask (Class)**

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

#### **7.3.1.1.30 MoveToPresetCmd (Class)**

This class represents the information needed to create a move to preset command to be added on the CommandQueue.

#### **7.3.1.1.31 NTCIPCamera (Class)**

This interface is used to represent an NTCIP model video camera in the field. The system contains an instance of this interface for each NTCIP video camera.

#### **7.3.1.1.32 NTCIPCameraCommands (Class)**

This class holds the ntcip command OIDs so the mib db does not have to be queried after startup.

#### **7.3.1.1.33 NTCIPCameraConfig (Class)**

This structure defines configuration data for the NTCIP type video camera.

#### **7.3.1.1.34 NTCIPCameraImpl (Class)**

This class implements the NTCIPCamera interface, and inherits from the ControllableCameraImpl class. The NTCIPCameraImpl implements methods of NTCIPCamera, extending the controllable camera to include NTCIP-specific operations. This class will contain a configuration and status object as necessary to convey NTCIP-specific configuration and status information.



#### **7.3.1.1.35 NTCIPCameraPositionReference (Class)**

This class represents the NTCIP protocol Camera Position Reference object. This object is used in position commands to configure the speed and direction of movement.

#### **7.3.1.1.36 NTCIPCameraProtocolHdlr (Class)**

This object contains the protocol for communication with a NTCIP Camera.

#### **7.3.1.1.37 NTCIPCameraStatus (Class)**

This structure defines the status data for the NTCIP video camera type.

#### **7.3.1.1.38 PollCameraTask (Class)**

The PollCameraTask is a timer task. When the timer fires it polls a camera by sending a poll command to the camera.

#### **7.3.1.1.39 PushEventSupplier (Class)**

This class provides a utility for application modules that push events on an event channel. The user of this class can pass a reference to the event channel factory to this object. The constructor will create a channel in the factory. The push method is used to push data on the event channel. The push method is able to detect if the event channel or its associated objects have crashed. When this occurs, a flag is set, causing the push method to attempt to reconnect the next time push is called. To avoid a supplier with a heavy supply load from causing reconnect attempts to occur too frequently, a maximum reconnect interval is used. This interval specifies the quickest reconnect interval that can be used. The push method uses this interval and the current time to determine if a reconnect should be attempted, thus reconnects can be throttled independently of a supplier's push rate.

#### **7.3.1.1.40 PutCameraOnlineCmd (Class)**

This class represents the information needed to request a put camera online command to be added on the CommandQueue.

#### **7.3.1.1.41 QueueableCommand (Class)**

A QueueableCommand is an interface used to represent a command that can be placed on a CommandQueue for asynchronous execution. Derived classes implement the execute method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a CommandQueue. The CommandQueue driver calls the execute method to execute a command in the queue and a call to the interrupted method is made when a CommandQueue is shut down.

#### **7.3.1.1.42 RequestCameraControlCmd (Class)**

This class represents the information needed to request a camera control command to be added on the CommandQueue.

#### **7.3.1.1.43 RequestCameraOverrideCmd (Class)**

This class represents the information needed to request a camera control override command to be added on the CommandQueue.

#### **7.3.1.1.44 RevokeControlCmd (Class)**

This class represents the information needed to create a revoke camera control command to be added on the CommandQueue.

#### **7.3.1.1.45 RevokeDisplayCmd (Class)**

This class represents the information needed to create a revoke camera display command to be added on the CommandQueue.

#### **7.3.1.1.46 ServiceApplication (Class)**

This interface is implemented by objects that can provide the basic services needed by a ChartII service application. These services include providing access to basic CORBA objects that are needed by service applications, such as the ORB, POA, Trader, and Event Service.

#### **7.3.1.1.47 ServiceApplicationModule (Class)**

This interface is implemented by modules that serve CORBA objects. Implementing classes are notified when their host service is initialized and when it is shutdown. The implementing class can use these notifications along with the services provided by the invoking ServiceApplication to perform actions such as object creation and publication.

#### **7.3.1.1.48 TakeCameraOfflineCmd (Class)**

This class represents the information needed to request a take camera offline command to be added on the CommandQueue.

#### **7.3.1.1.49 TerminateControlCmd (Class)**

This class represents the information needed to request a terminate camera control command to be added on the CommandQueue.

#### **7.3.1.1.50 UniquelyIdentifiable (Class)**

This interface will be implemented by all classes which are to be identifiable within the system. The identifier must be generated by the IdentifierGenerator to ensure uniqueness.

#### **7.3.1.1.51 ViconSVFTCameraImpl (Class)**

This class implements the ViconSVFTCamera interface, and inherits from the ControllableCameraImpl class. The ViconSurveyorVFTCameraImpl implements methods of ViconSVFTCamera, extending the controllable camera to include Vicon SVFT-specific operations. This class will contain a configuration and status object as necessary to convey Vicon SVFT-specific configuration and status information.

#### **7.3.1.1.52 ViconSVFTCameraStatus (Class)**

The ViconSVFTCameraStatus class is used to hold camera status information at the ViconSVFTCamera level. Only ViconSVFTCamera specific information is stored.

#### **7.3.1.1.53 ViconSVFTProtocolHdlr (Class)**

This class contains an implementation for Vicon SVFT camera control commands. It translates every camera command (pan, tilt, zoom...) into bytes that a Vicon SVFT camera understands. Then, it uses a CameraControlDevice to send the byte codes to the camera and evaluate responses from the camera.

#### **7.3.1.1.54 VideoCamera (Class)**

The VideoCamera interface is implemented by objects representing controllable video cameras within the CHART II system. The VideoCamera interface represents a controllable video camera as opposed to the uncontrollable, immovable FixedVideoCamera, the other type of GenericVideoCamera. (The VideoCamera class could have been called the ControllableVideoCamera interface, but since the CHART II video system exists primarily to control controllable video cameras, the camera hierarchy has been arranged to avoid the longish name ControllableVideoCamera.) Current plans call for classes to represent a COHU MPC camera, COHU 3955 camera, Vicon SVFT camera, and NTCIP-compliant camera, and there are interfaces defined for each of these subtypes of VideoCamera. The VideoCamera interface includes the GeoLocatable interface, to someday allow for advanced features such as automatic identification of cameras near traffic events, automatic pointing of cameras to traffic events, etc.

The VideoCamera interface includes all methods common to the three known types of video cameras currently in use by MDSHA, although it is likely to contain a superset of methods which would be implemented by the entire universe of all video cameras which could someday be used. This interface may have to be refined in the event that future brands or models of video cameras might be incorporated under CHART II, but it is an appropriate set of methods for the present day.

#### **7.3.1.1.55 VideoCameraConfig (Class)**

The VideoCameraConfig structure is used to hold configuration information about VideoCamera objects at the VideoCamera level. Further details about lower-level VideoCamera subclasses are provided by subclasses of VideoCameraConfig.

#### **7.3.1.1.56 VideoCameraFactory (Class)**

The VideoCameraFactory interface is implemented by factory classes responsible for creating, maintaining, and controlling a collection of VideoCamera objects.

#### **7.3.1.1.57 VideoCameraImpl (Class)**

The CameraImpl class provides an implementation of the VideoCamera interface, and by extension the VideoSource, SharedResource, CommEnabled, GeoLocatable, and UniquelyIdentifiable interfaces, as specified by the IDL.

This class contains a CommandQueue object that is used to sequentially execute long running operations in a thread separate from the CORBA request threads, thus allowing quick initial responses.

Also contained in this class are VideoCameraConfig and VideoCameraStatus objects (used to store the configuration and status of the camera), and a VideoCameraData object (used to store internal status information which is persisted but not pushed out to clients).

The CameraImpl contains \*Impl methods that map to methods specified in the IDL, including requests to display the camera video on a monitor, remove the camera video from a monitor, put the camera online, put the camera offline, put the camera in maintenance mode (future), or to change (set) the configuration of the camera (future). Some of these requests require (or potentially require) field communications to the device, so each request is stored in a specific subclass of QueueableCommand and added to the CommandQueue. The queueable command objects simply call the appropriate CameraImpl method as the command is executed by the CommandQueue in its thread of execution.

The CameraImpl also contains methods called by the CameraFactory to support the timer tasks of the Camera Service: to look for Cameras with no one logged in at the controlling operations center, and to initiate recovery processing if needed (future).

#### **7.3.1.1.58 VideoCameraStatus (Class)**

The VideoCameraStatus structure is used to hold status information about VideoCamera objects at the VideoCamera level. Further details about lower-level VideoCamera subclasses are provided by subclasses of VideoCameraStatus.

#### **7.3.1.1.59 VideoProvider (Class)**

The VideoProvider interface is a generic abstract interface including VideoSource objects (e.g. video cameras) and BridgeCircuit objects. Both VideoSource and BridgeCircuit objects provide video to a VideoCollector, but only VideoSource objects are true origins of video which a typical user would have direct interaction with. BridgeCircuit VideoProvider objects merely pass on video provided from elsewhere in a VideoRoute.

#### **7.3.1.1.60 VideoProviderConfig (Class)**

This structure defines configuration data common to all video sources.

#### **7.3.1.1.61 VideoProviderImpl (Class)**

This class implements the VideoProvider interface as an abstract class. Subclasses for this class are the VideoCameraImpl and BridgeCircuitProviderImpl class.

#### **7.3.1.1.62 VideoProviderStatus (Class)**

The VideoProviderStatus structure is used to hold and transmit status information about VideoProvider objects at the VideoProvider level. Further details about lower-level VideoProvider subclasses are provided by subclasses of VideoProviderStatus.

#### **7.3.1.1.63 VideoSource (Class)**

The VideoSource interface is implemented by objects which originate video signals, such as video cameras and image generators. Within the user interface, the VideoSource interface represents all video sources which can be put on monitors (i.e., VideoSink objects).

The VideoSource interface includes the SharedResource interface. A VideoSource is controlled by an Operations Center if the VideoSource is in maintenance mode, or if the VideoSource is a camera which has an active control session up.

#### **7.3.1.1.64 VideoSourceConfig (Class)**

This structure defines configuration data common to all video sources.

#### **7.3.1.1.65 VideoSourceStatus (Class)**

The VideoSourceStatus structure is used to hold and transmit status information about VideoSource objects at the VideoSource level. Further details about lower-level VideoSource subclasses are provided by subclasses of VideoSourceStatus.

## 7.3.2 Sequence Diagrams

### 7.3.2.1 CameraControlModule:AddCamera (Sequence Diagram)

This sequence diagram shows the implementation of the createCamera interface of the VideoCameraFactory. There are actually three create methods in the factory, one for each type of camera: COHU 3955, Surveyor VFT and NTCIP. Since they all work the same way, they are all represented by createCamera(). First a check is performed to verify that the operator has sufficient privileges to create a camera. Next, the camera is inserted into the database. All video sending devices and flash video stream controls configurations are stored in the database. Part of this process includes creating the camera object itself. Finally, the new camera object is activated and the event is pushed out to clients.

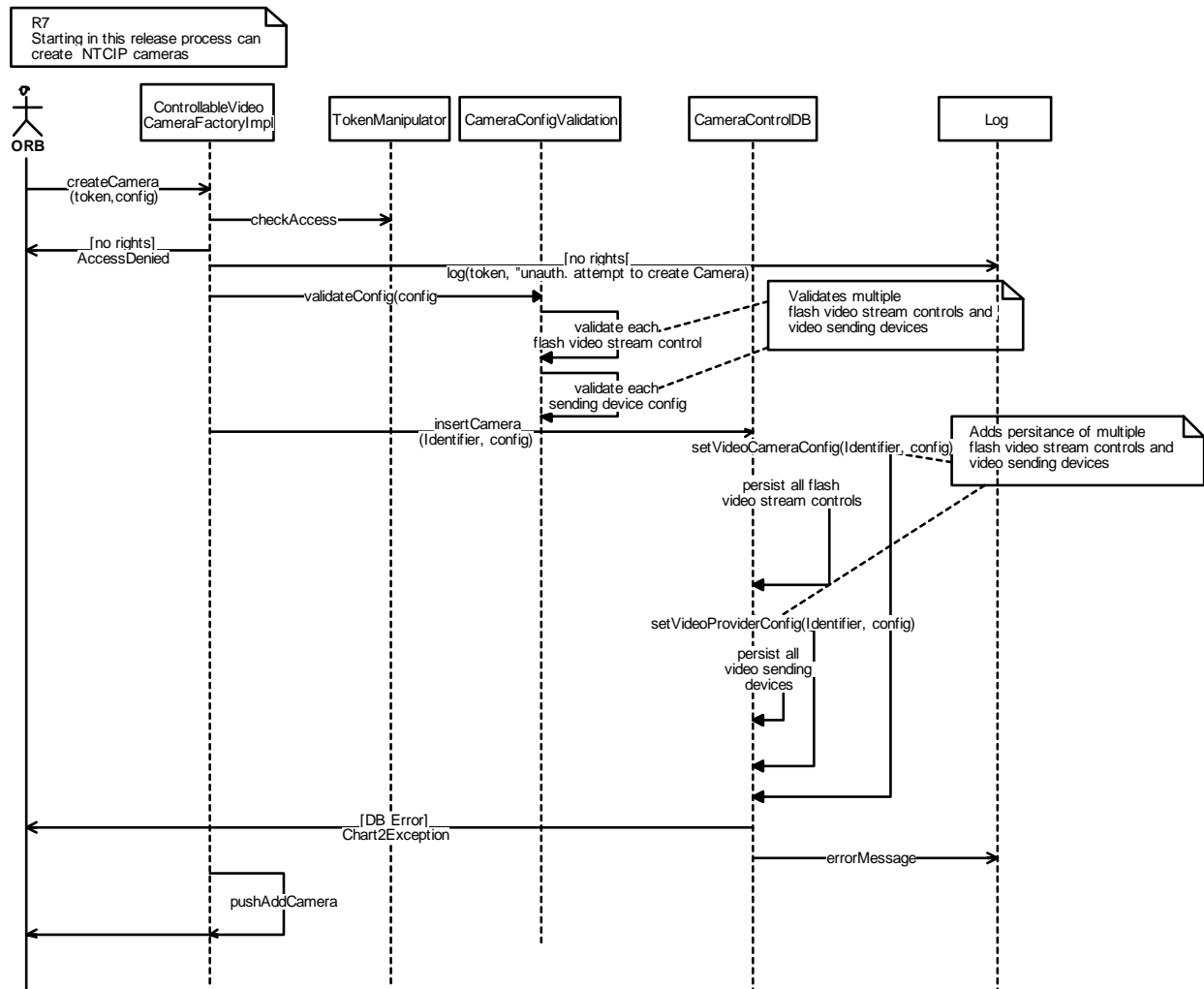
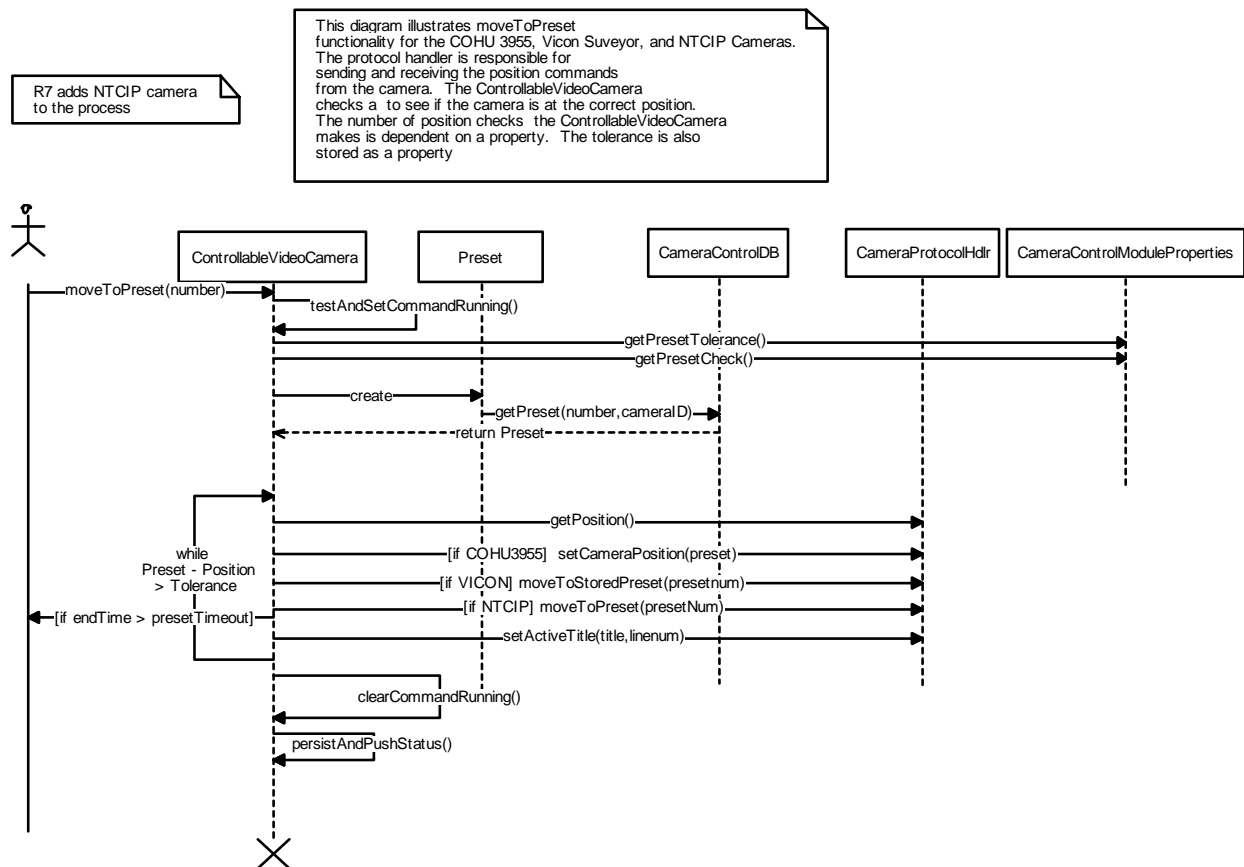


Figure 7-8. CameraControlModule:AddCamera (Sequence Diagram)

### 7.3.2.2 CameraControlModule:MoveToPreset (Sequence Diagram)

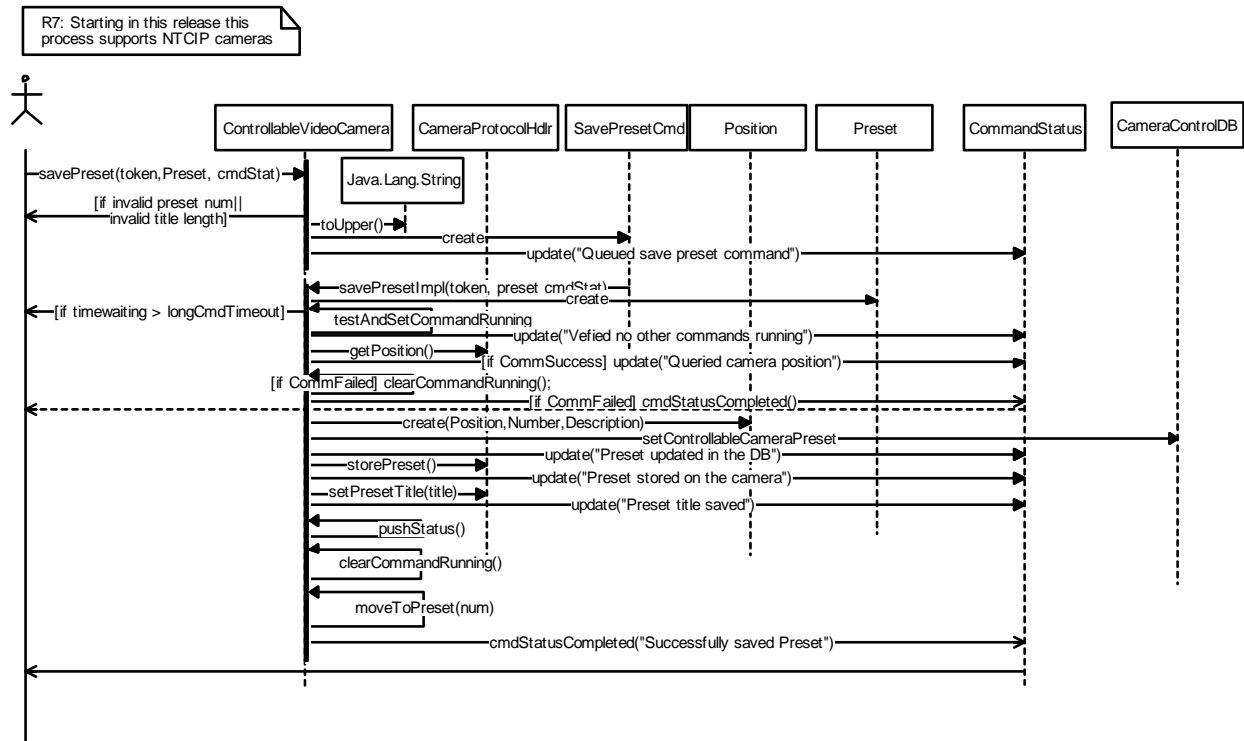
This sequence diagram describes the process of moving to a preset for COHU 3955, Vicon, and NTCIP cameras. The absolute position of the camera is retrieved and the command is built up and sent to the camera by the protocol handler. Next, the camera is queried to get its absolute position as it moves to the preset. Once it reaches the expected position (within a tolerance), the operation succeeds. Otherwise the operation fails.



**Figure 7-9. CameraControlModule:MoveToPreset (Sequence Diagram)**

### 7.3.2.3 CameraControlModule:SavePreset (Sequence Diagram)

This operation is used to save the current camera position as a preset.



**Figure 7-10. CameraControlModule:SavePreset (Sequence Diagram)**



### 7.3.2.4 CameraControlModule:SetCameraConfiguration (Sequence Diagram)

This sequence diagram shows the implementation of the setConfiguration interface of the CameraImpl class (which represents VideoProviderImpl, VideoSourceImpl, VideoCameraImpl etc.). First a check is performed to verify that the operator has sufficient privileges to update a camera. Next a check is made to see that the camera is offline. Only offline cameras may have their configurations updated. If the camera is offline, the new configuration is validated. During this process all flash video stream control and video sending device configurations are validated. Next the new configuration is written to the database. During the set configuration process, all flash video stream controls and video sending devices configured for the camera are updated in the database. Finally, the camera is apprised of its new configuration.

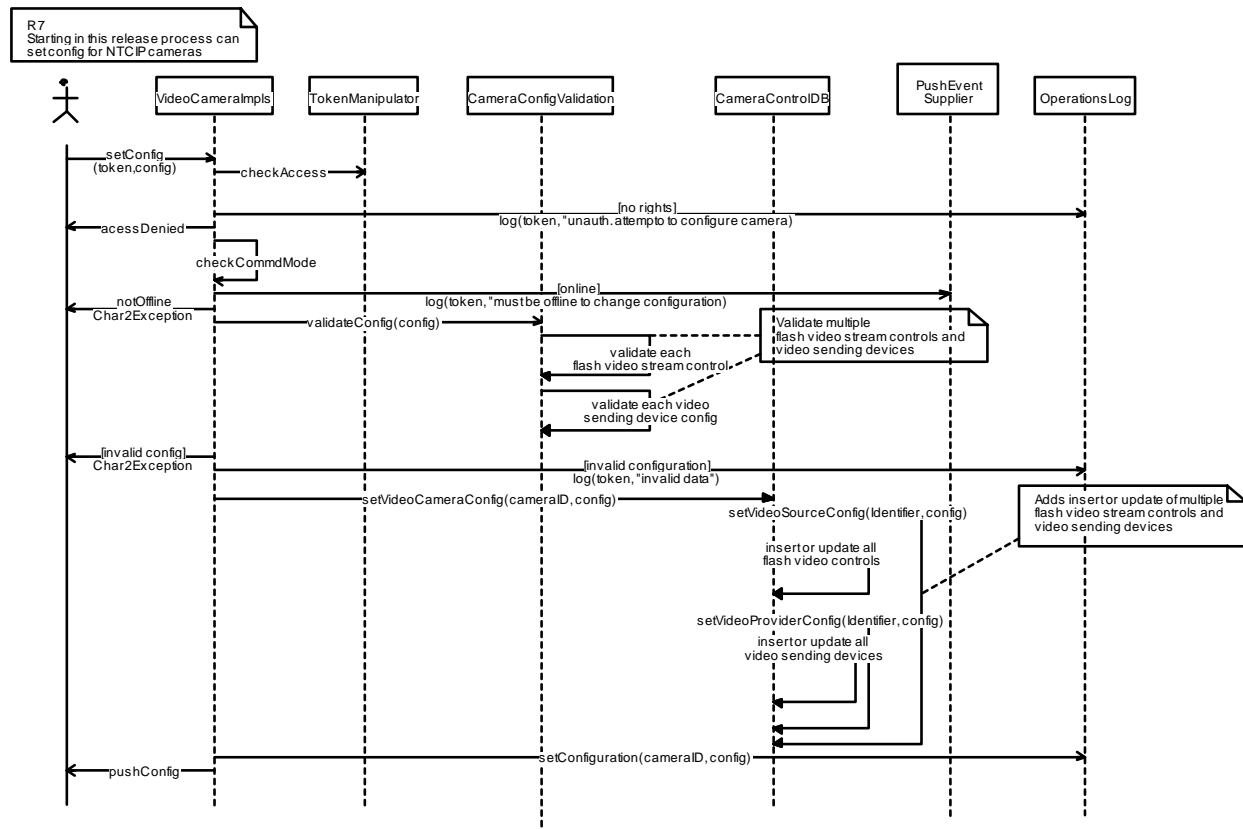


Figure 7-11. CameraControlModule:SetCameraConfiguration (Sequence Diagram)

### 7.3.2.5 CameraControlModule:TakeCameraOffline (Sequence Diagram)

This Sequence Diagram shows the process of taking a camera offline. This command is executed asynchronously to avoid having to block for commands that may take a long time.

The bulk of the work is done in the for loop with the cameraUnavailable. We check to see if the camera is being displayed on the monitor, if so, we call displayImage with the NoVideoAvailableSource. The camera status will be updated to show that it is offline. If this camera is part of a tour, the camera will be skipped since its status will be offline.

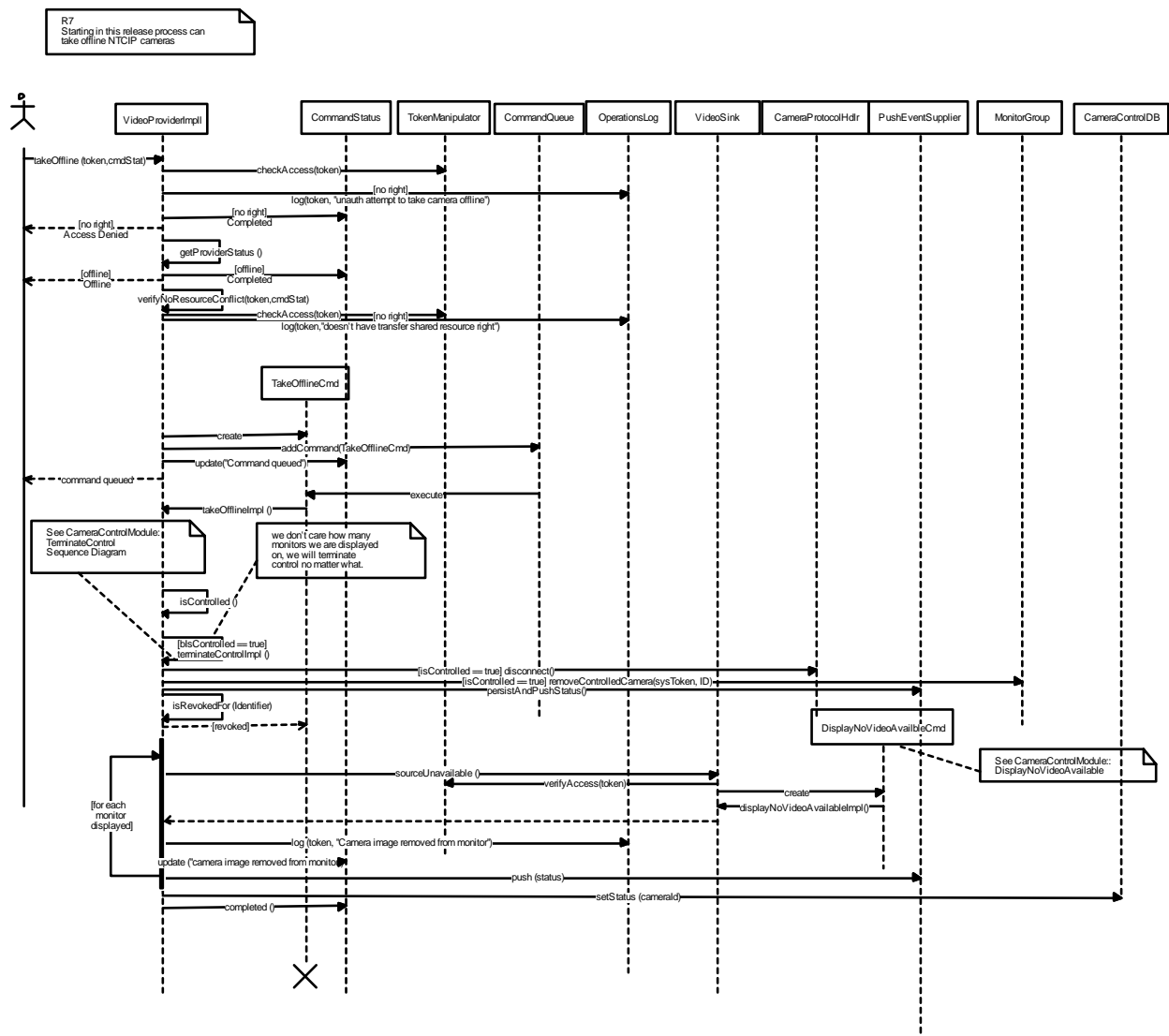


Figure 7-12. CameraControlModule:TakeCameraOffline (Sequence Diagram)

### 7.3.2.6 Encoder:receive (Sequence Diagram)

This diagram shows the processing that occurs when receiving data on an ip encoder port using initial,interchar,and maxDuration timeouts.

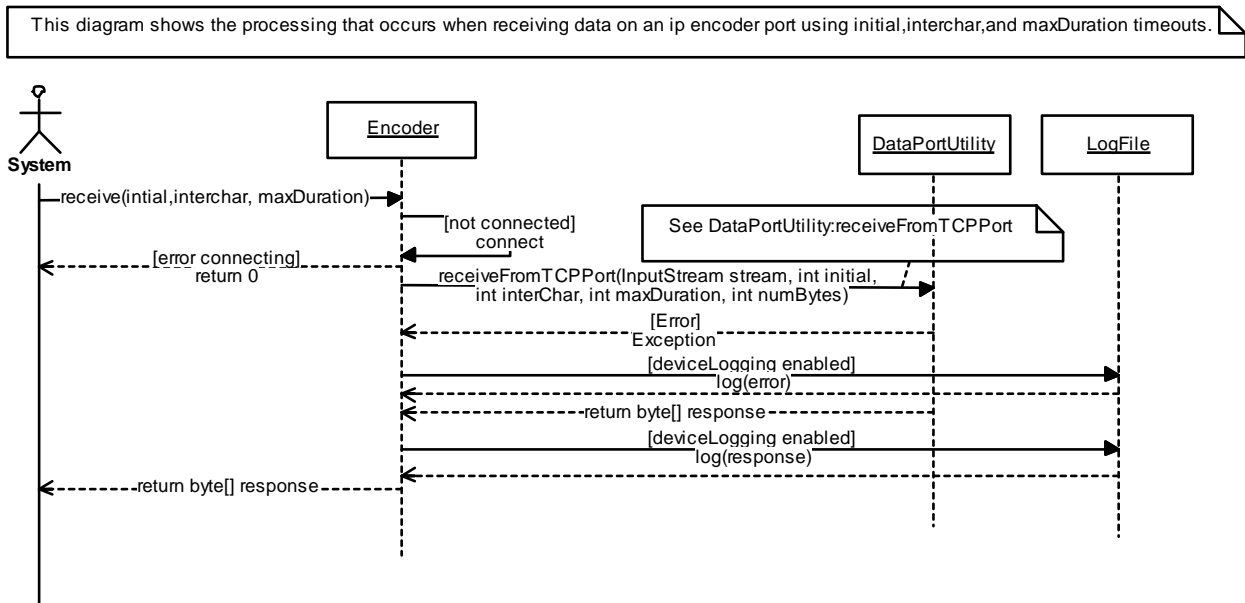
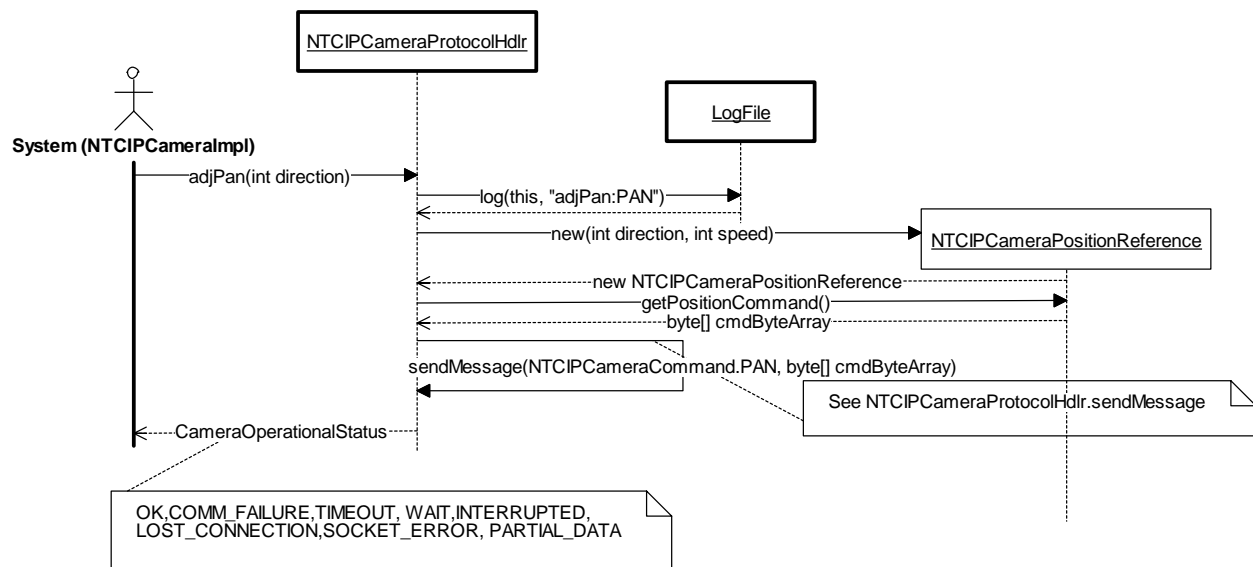


Figure 7-13. Encoder:receive (Sequence Diagram)

### 7.3.2.7 NTCIPCameraProtocolHdlr:adjPan (Sequence Diagram)

This diagram shows the processing that occurs when an adjust pan command is sent from the NTCIPCameraImpl to the NTCIPCameraProtocolHdlr. The protocol hdlr contains a set of NTCIPCameraCommands consisting of all command OID's at construction time. If the control device is an encoder, the command is sent through a shared NTCIP utility class, otherwise the existing CameraControlDevice code is used for data transmission.

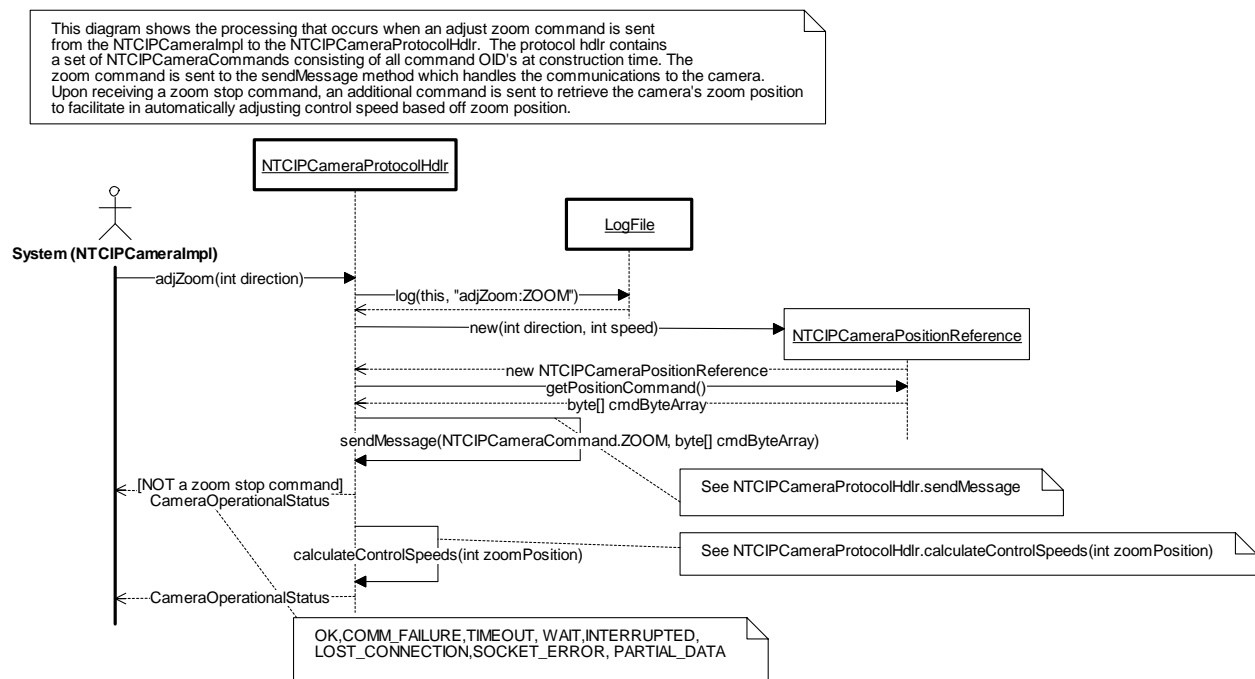
This diagram shows the processing that occurs when an adjust pan command is sent from the NTCIPCameraImpl to the NTCIPCameraProtocolHdlr. The protocol hdlr contains a set of NTCIPCameraCommands consisting of all command OID's at construction time. The pan command is sent to the sendMessage method which handles the communications to the camera.



**Figure 7-14. NTCIPCameraProtocolHdlr:adjPan (Sequence Diagram)**

### 7.3.2.8 NTCIPCameraProtocolHdlr:adjZoom (Sequence Diagram)

This diagram shows the processing that occurs when an adjust zoom command is sent from the NTCIPCameraImpl to the NTCIPCameraProtocolHdlr. The protocol hdlr contains a set of NTCIPCameraCommands consisting of all command OID's at construction time. The zoom command is sent to the sendMessage method which handles the communications to the camera. Upon receiving a zoom stop command, an additional command is sent to retrieve the camera's zoom position to facilitate in automatically adjusting control speed based off zoom position.



**Figure 7-15. NTCIPCameraProtocolHdlr:adjZoom (Sequence Diagram)**

### 7.3.2.9 NTCIPCameraProtocolHdlr:calculateControlSpeeds (Sequence Diagram)

This method shows the processing that occurs to adjust the pan and tilt control speeds based off the zoom position. The camera configuration will store min/max zoom positions and a min/max pan/tilt control speeds.

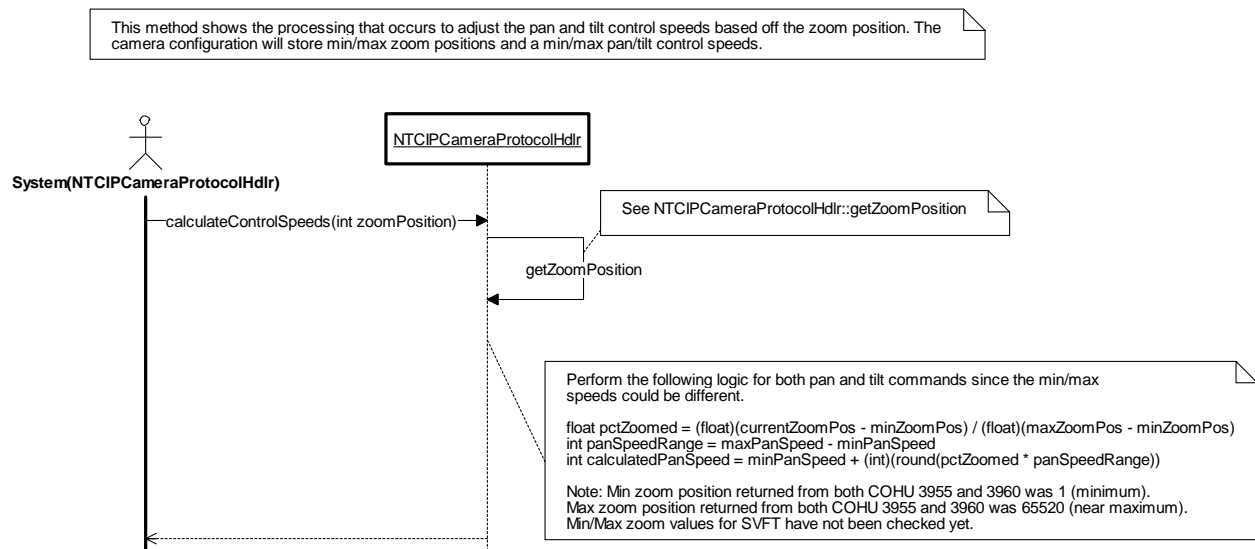
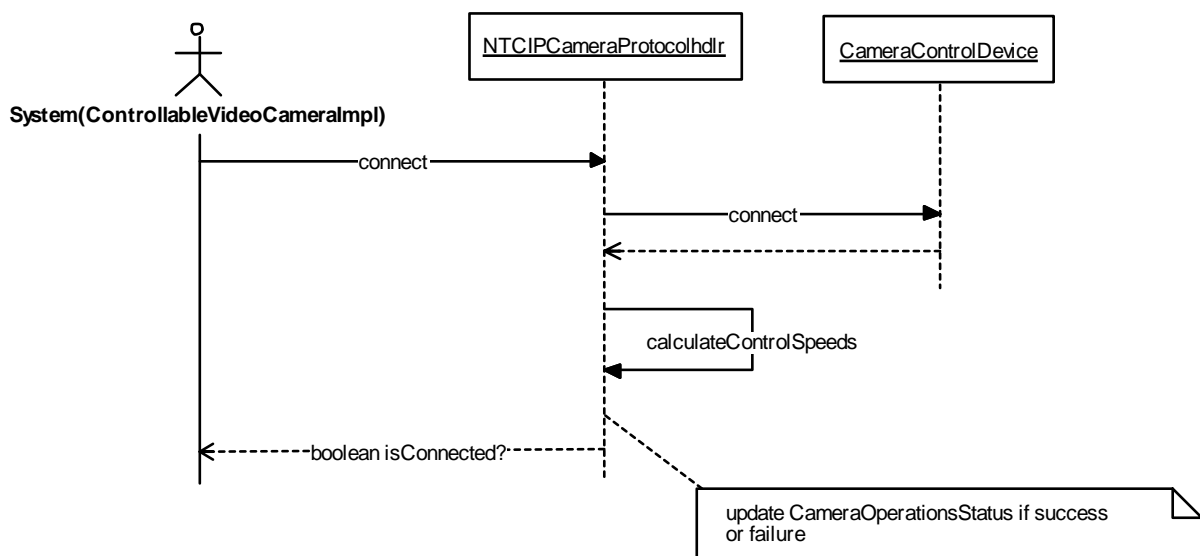


Figure 7-16. NTCIPCameraProtocolHdlr:calculateControlSpeeds (Sequence Diagram)

### 7.3.2.10 NTCIPCameraProtocolHdlr:connect (Sequence Diagram)

This diagram shows the processing that occurs when connecting to an NTCIP camera for a control session. Upon connecting, the zoom position is retrieved to ensure the control speed is scaled correctly.

This diagram shows the processing that occurs to open a connection to an NTCIP camera. This occurs when a control session is opened and from a poll. When an NTCIP camera control session is opened, a command is sent to query the camera's current zoom position, and adjust the control speed as necessary.



**Figure 7-17. NTCIPCameraProtocolHdlr:connect (Sequence Diagram)**

### 7.3.2.11 NTCIPCameraProtocolHdlr:getZoomPosition (Sequence Diagram)

This diagram shows the processing that occurs when a request for the camera's zoom position is requested. This will occur when a control session is first opened and also from within a zoom stop. This method will first try to use a specific mib to query the zoom position, If that does not work the mib for setting the camera zoom will be used to determine the zoom position. If using the later, the response will have to be parsed from a hex string to determine the position.

This diagram shows the processing that occurs when a request for the camera's zoom position is requested. This will occur when a control session is first opened and also from within a zoom stop. This method will first try to use a specific MIB to query the zoom position, If that does not work the MIB for setting the camera zoom will be used to determine the zoom position. If using the latter, the response will be parsed from a hex string to determine the position.

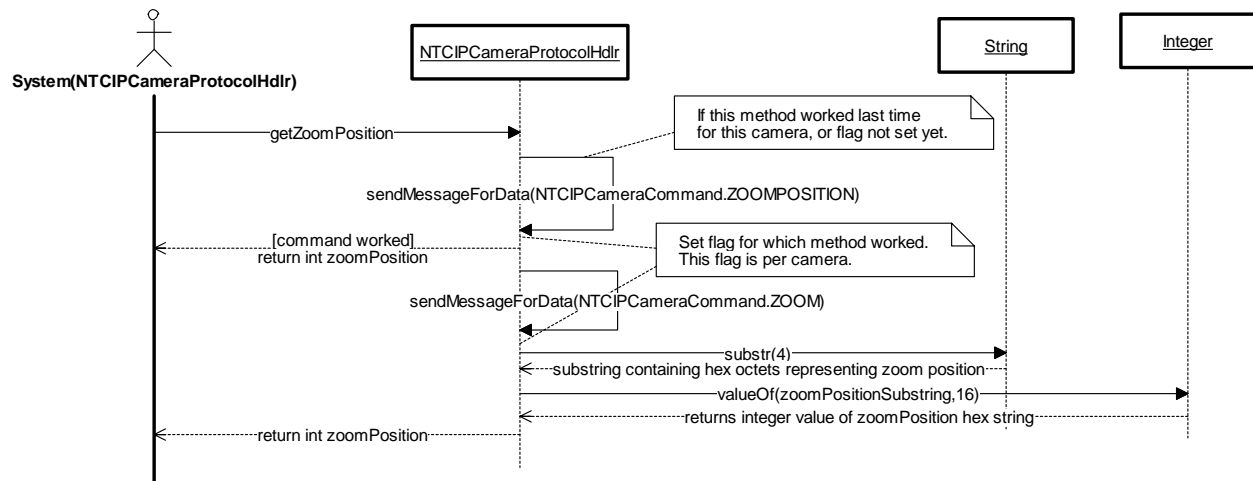


Figure 7-18. NTCIPCameraProtocolHdlr:getZoomPosition (Sequence Diagram)



### 7.3.2.12 NTCIPCameraProtocolHdlr:moveToPreset (Sequence Diagram)

This diagram shows the processing that occurs when an NTCIP camera is moved to a stored preset position. The camera is moved to the stored preset. After moving, a command is sent to enable the preset title for the 2nd line.

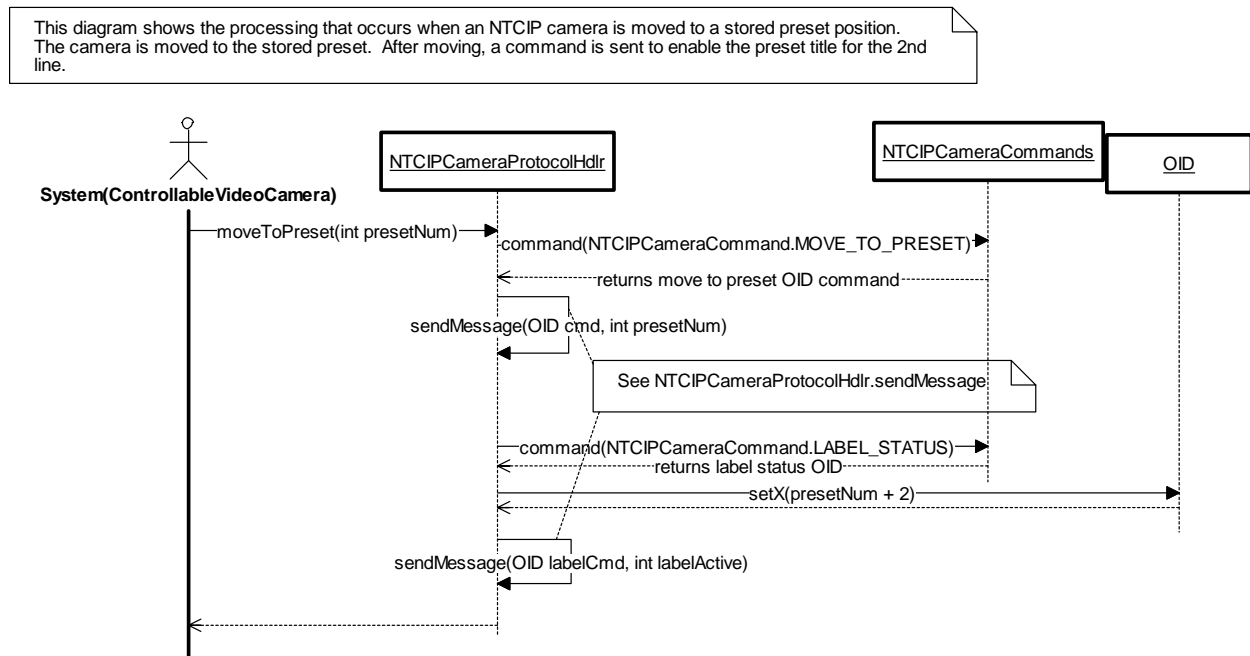
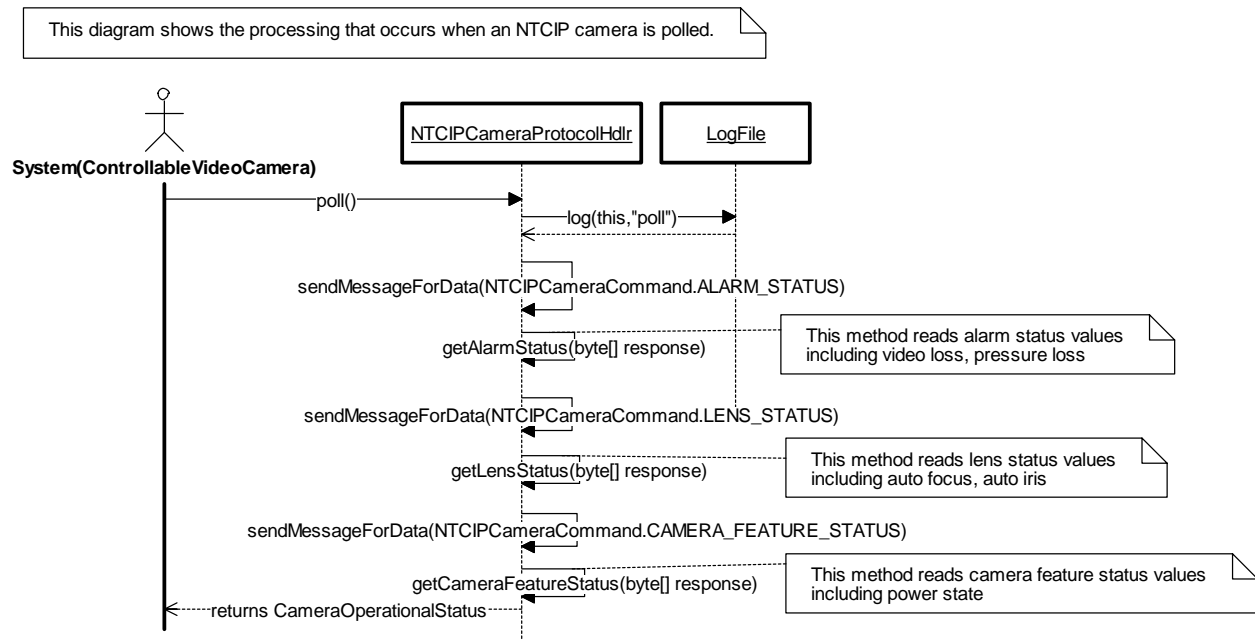


Figure 7-19. NTCIPCameraProtocolHdlr:moveToPreset (Sequence Diagram)

### 7.3.2.13 NTCIPCameraProtocolHdlr:poll (Sequence Diagram)

This diagram shows the processing that occurs when an ntcip camera is polled.



**Figure 7-20. NTCIPCameraProtocolHdlr:poll (Sequence Diagram)**

### 7.3.2.14 NTCIPCameraProtocolHdlr:sendMessage (Sequence Diagram)

This diagram shows the processing that occurs when a command is sent to an NTCIP camera. The command is looked up from a collection stored in the protocol hdlr in the form of an OID. The command datapayload is passed and set in the command. A DataPortUtility object is created that sets either an encoder or CameraControlComPort as the control device. The DataPortUtility class is then passed to the NTCIPUtility class along with the OID to send the command to the camera.

This diagram shows the processing that occurs when a command is sent to an NTCIP camera. The command is looked up from a collection stored in the protocol hdlr in the form of an OID. The command datapayload is passed and set in the command. A DataPortUtility object is created that sets either an encoder or CameraControlComPort as the control device. The DataPortUtility class is then passed to the NTCIPUtility class along with the OID to send the command to the camera.

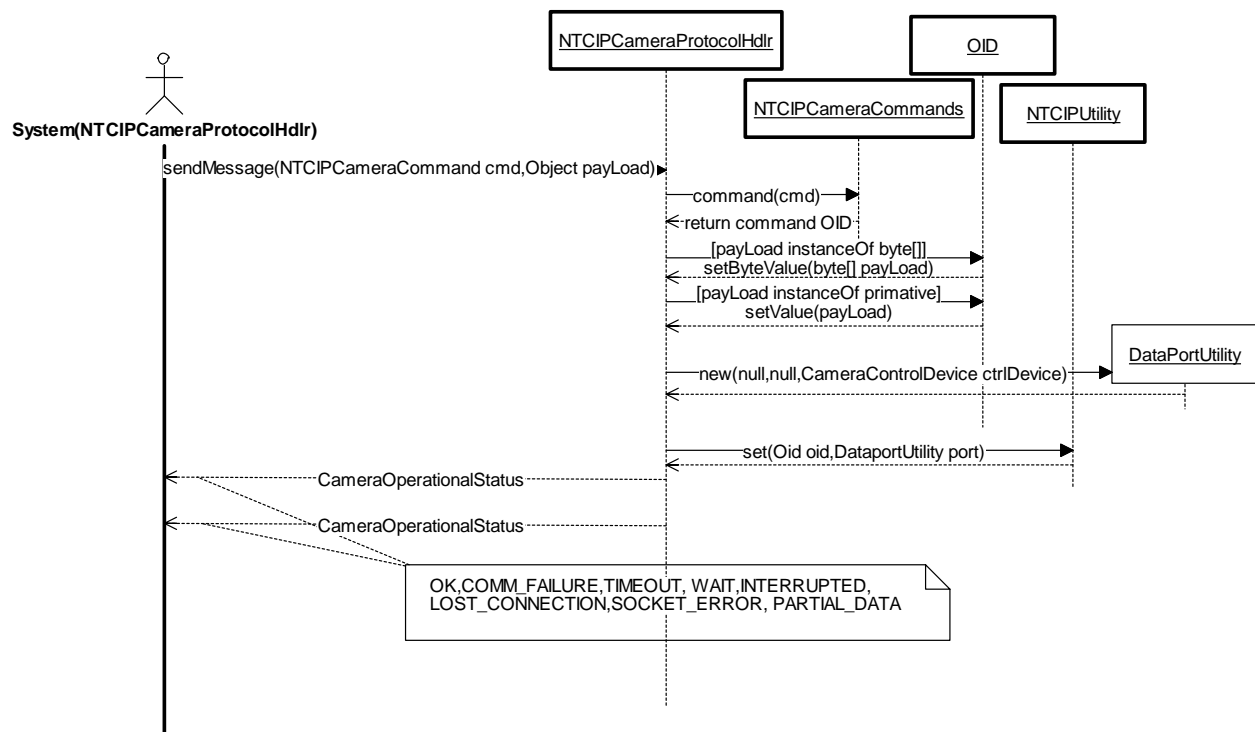


Figure 7-21. NTCIPCameraProtocolHdlr:sendMessage (Sequence Diagram)

### 7.3.2.15 NTCIPCameraProtocolHdlr:sendMessageForData (Sequence Diagram)

This diagram shows the processing that occurs when a command is sent to an NTCIP camera where a response is expected. The command is looked up in the protocolHdlr's stored array of commands. If a payload is specified, the value is updated in the OID. A DataPortUtility is created and passed along with the OID to the NTCIPUtility which executes the command and returns the response.

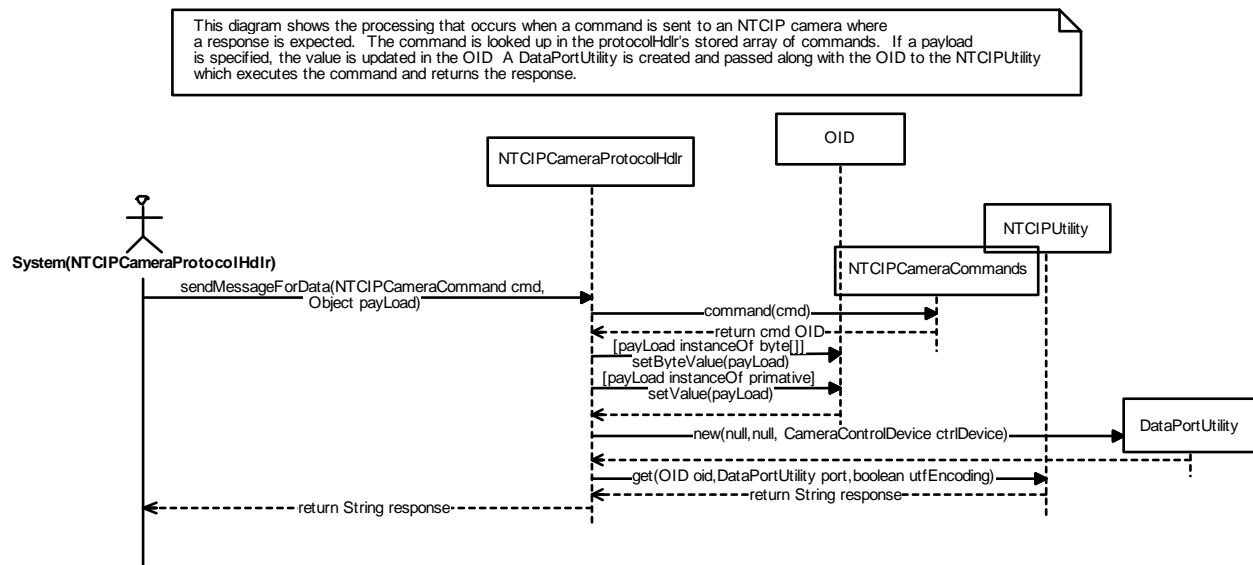


Figure 7-22. NTCIPCameraProtocolHdlr:sendMessageForData (Sequence Diagram)

### 7.3.2.16 NTCIPCameraProtocolHdlr:setLabelText (Sequence Diagram)

This diagram shows the processing that occurs when setting a label/title. One command sets the text of the label. The second command activates the label.

This diagram shows the processing that occurs when setting a label/title. One command sets the text of the label. The second command activates the label.

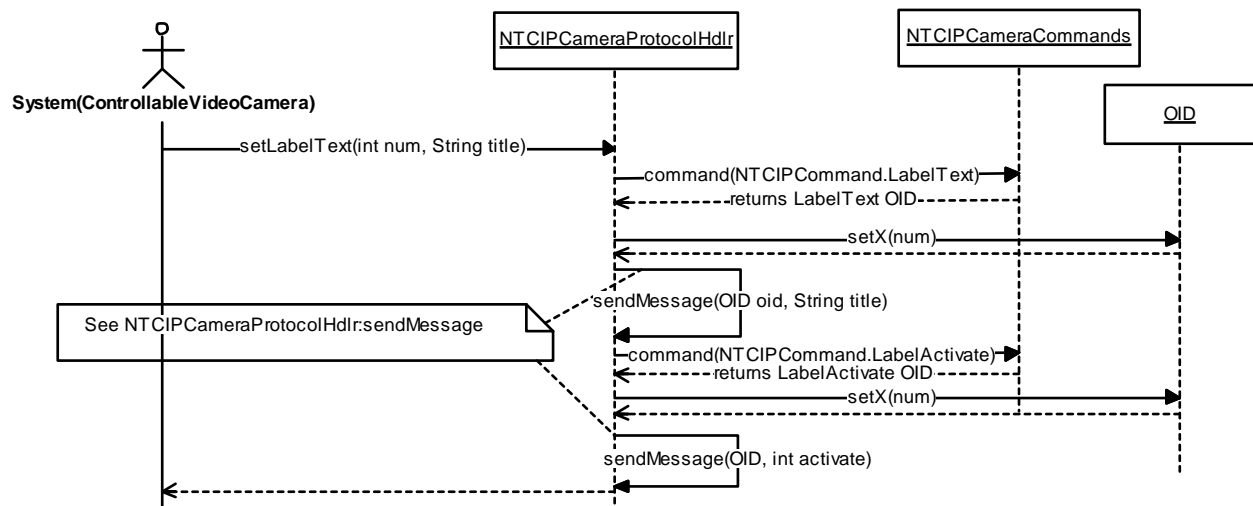
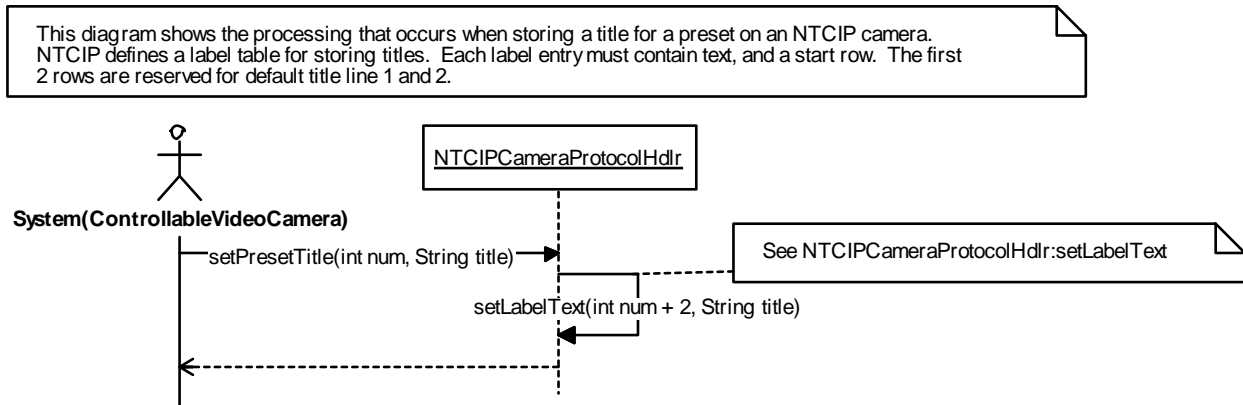


Figure 7-23. NTCIPCameraProtocolHdlr:setLabelText (Sequence Diagram)

### 7.3.2.17 NTCIPCameraProtocolHdlr:setPresetTitle (Sequence Diagram)

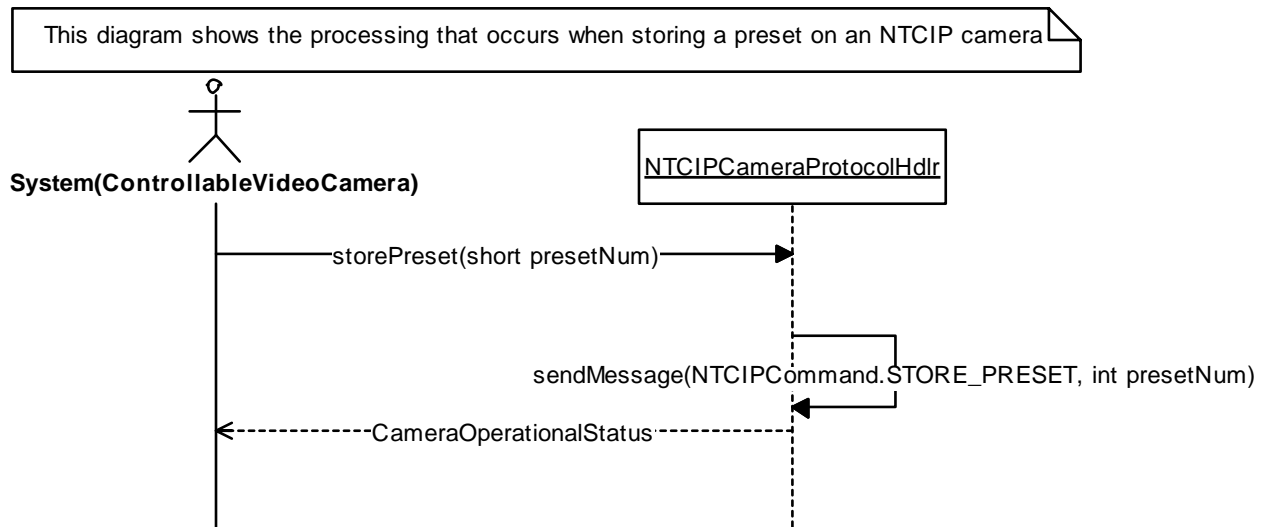
This diagram shows the processing that occurs when storing a title for a preset on an NTCIP camera. NTCIP defines a label table for storing titles. Each label entry must contain text, and a start row. The first two rows are reserved for default title line 1 and 2.



**Figure 7-24. NTCIPCameraProtocolHdlr:setPresetTitle (Sequence Diagram)**

### 7.3.2.18 NTCIPCameraProtocolhdlr:storePreset (Sequence Diagram)

This diagram shows the processing that occurs when storing a preset on an NTCIP camera



**Figure 7-25. NTCIPCameraProtocolhdlr:storePreset (Sequence Diagram)**

#### 7.4.1.1 CameraNTCIPComplianceTesterClasses (Class Diagram)



This interface is implemented by objects that wish to be notified when the user has requested to exit the application. This interface was introduced to keep the main application class from having to implement the `awt` action listener and window listener



interfaces, most of which do not apply to the main application class (it just needs to know when the user wants to close).

#### **7.4.1.1.2 CameraAsyncCommandExecuter (Class)**

This class is a queueable command used to execute a Camera command asynchronous to the main GUI thread, allowing the GUI to process events as a test is running. When the command is run, it notifies the test activation listener based on the command type that was specified during construction.

#### **7.4.1.1.3 CameraCommandType (Class)**

This is an enumeration of the types of commands that can be tested.

#### **7.4.1.1.4 CameraCommSettings (Class)**

This class holds communication related settings for the CameraNTCIPComplianceTester. The settings are persisted in a .props file and are loaded on construction (or set to default values if .props file doesn't yet exist). The save method saves the settings to a .props file. Getters and Setters exist for each of the members in this class.

#### **7.4.1.1.5 CameraCommSettingsDlg (Class)**

This class is a dialog that allows the user to modify and save the communications settings used by the compliance tester.

#### **7.4.1.1.6 CameraNTCIPComplianceTester (Class)**

This class contains the main entry point for the NTCIP Camera Compliance tester. Its main method instantiates an instance of the class, whose constructor initializes the application. Initialization includes initializing the ORB and POA, instantiating the various setting objects (which depersist their settings from props files), creating a CameraTestRunner object (which executes the actual tests on command), and creates the main window used to interact with the application.

#### **7.4.1.1.7 CameraNTCIPProtocolHdlrConfig (Class)**

This class contains configuration values specific to the NTCIPCameraProtocolHdlr.

#### **7.4.1.1.8 CameraNTCIPTesterMainWindow (Class)**

This class is the main window for the NTCIP Camera Compliance Tester. It has a JFrame which it populates with various GUI objects, such as a menu bar with menu items, and a scroll pane with a text area so it can show test results. It implements the ActionListener and WindowListener interfaces and handles events for each menu click in addition to the window closing event that is fired if the user closes the window using the X.

#### **7.4.1.1.9 CameraTestActivationListener (Class)**

This interface specifies methods to be implemented by objects that are to be notified when the user activates a test.

#### **7.4.1.1.10 CameraTestResultRecord (Class)**

This interface specifies methods to be implemented by objects that can record the results of tests.

#### **7.4.1.1.11 CameraTestRunner (Class)**

This class provides the capability to execute a test. It is notified when it is time to run a test through the TestActivationListener interface, and records all results to its associated CameraTestResultRecorder. This class makes use of existing CHART communications and protocol handler classes to ensure its tests are using the exact code being used by the CHART system to perform this functionality. There is no CHART business logic within this class, it is simply a controller that creates a communications port and passes it to the CHART protocol handler to perform the requested command.

#### **7.4.1.1.12 CommandQueue (Class)**

The CommandQueue class provides a queue for QueueableCommand objects. The CommandQueue has a thread that it uses to process each QueueableCommand in a first in first out order. As each command object is pulled off the queue by the CommandQueue's thread, the command object's execute method is called, at which time the command performs its intended task.

#### **7.4.1.1.13 CommPortConfig (Class)**

This structure is used to pass comm port configuration values during a connection request.

#### **7.4.1.1.14 DataPortUtility (Class)**

This class is a wrapper used to hide the underlying port being used to communicate (tcp/ip, FMS, or CameraControlDevice port).

#### **7.4.1.1.15 DirectPortImpl (Class)**

This class implements the DirectPort interface as defined in the IDL. Its connect method opens a javax.comm.SerialPort object and sets the port settings according to the baud, data bits, stop bits, and parity that was passed. Its disconnect method closes the javax.comm.SerialPort. This class also implements the send and receive functions as specified in the DataPort IDL interface. The send and receive methods use the read and write methods of the javax.comm.SerialPort object to send and receive bytes on the com port. While the send method contains little processing other than calling the javax.comm.SerialPort object's write method, the receive method contains logic that allows it to receive a burst of bytes before returning. This causes the receive method to return all available bytes on the port and thus helps to prevent the need for multiple calls to receive

for a single command response. This class updates a timestamp each time send or receive is called. When its `isInactive()` method is called, it checks the current time vs. the last send/receive time and if the difference is greater than the current inactivity timeout, it returns true.

#### **7.4.1.1.16 java.awt.event.ActionListener (Class)**

This interface listens for actions such as when a menu item or button is clicked. For menu items, it is attached to menu items when the menu is built.

#### **7.4.1.1.17 java.awt.event.WindowListener (Class)**

Listener interface that a class must implement for receiving window events

#### **7.4.1.1.18 javax.swing.JDialog (Class)**

This class is part of the JDK and provides functionality for dialog windows.

#### **7.4.1.1.19 javax.swing.JFileChooser (Class)**

This class is part of the JDK and provides functionality to allow the user to choose a file from their local file system.

#### **7.4.1.1.20 javax.swing.JFrame (Class)**

Java class that displays a frame window.

#### **7.4.1.1.21 NTCIPCameraProtocolHandler (Class)**

This object contains the protocol for communication with a NTCIP Camera.

#### **7.4.1.1.22 ORB (Class)**

The CORBA ORB (Object Request Broker) provides a common object oriented, remote procedure call mechanism for inter-process communication. The ORB is the basic mechanism by which client applications send requests to server applications and receive responses to those requests from servers.

#### **7.4.1.1.23 QueueableCommand (Class)**

A `QueueableCommand` is an interface used to represent a command that can be placed on a `CommandQueue` for asynchronous execution. Derived classes implement the `execute` method to specify the actions taken by the command when it is executed. This interface must be implemented by any device command in order that it may be queued on a `CommandQueue`. The `CommandQueue` driver calls the `execute` method to execute a command in the queue and a call to the `interrupted` method is made when a `CommandQueue` is shut down.

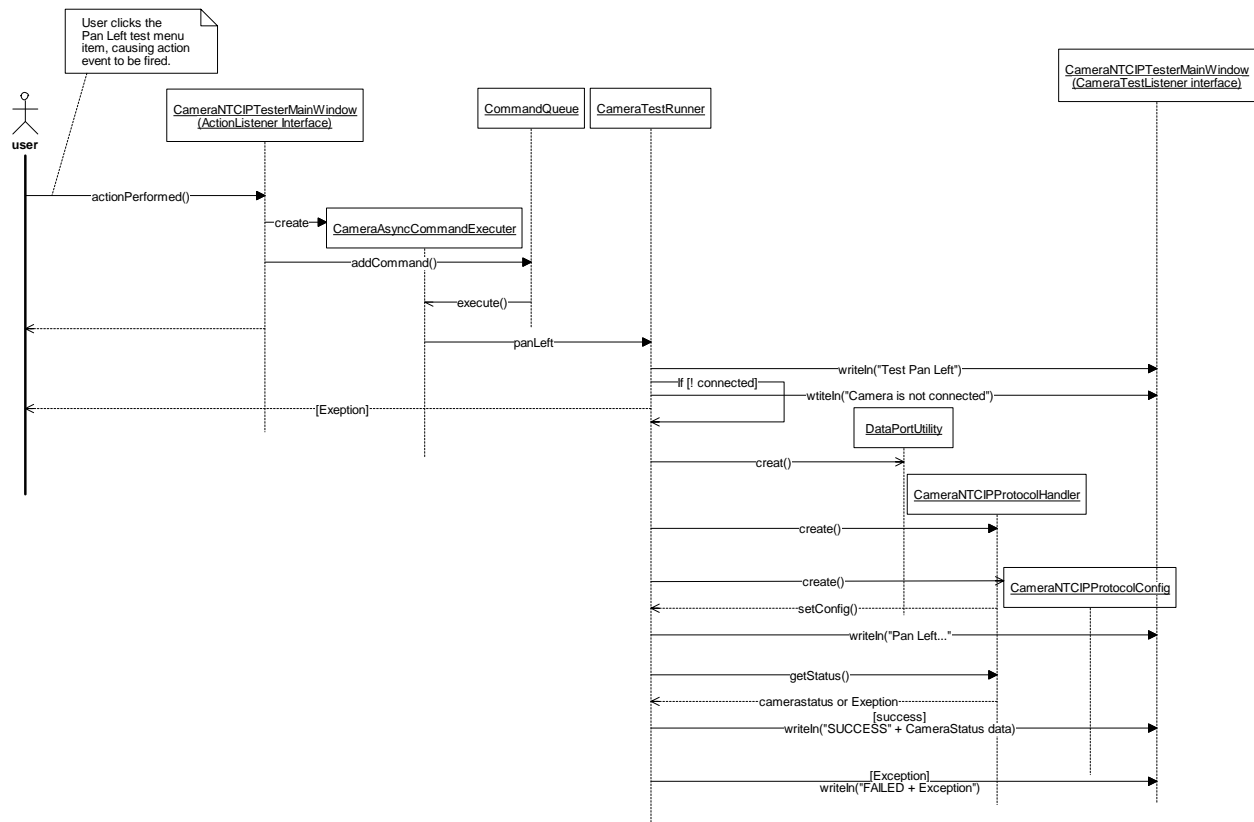
#### **7.4.1.1.24 TCPPort (Class)**

This class provides access to a TCP/IP port for device communications.

## 7.4.2 Sequence Diagrams

### 7.4.2.1 CameraNTCIPComplianceTester:PanLeft (Sequence Diagram)

This diagram shows the processing that takes place when the user chooses to execute the pan left test. This sequence is prototypical of all tests that may be executed, with slight variations as pointed out below. When the user clicks one of the test menu items, the main window is notified via its ActionListener interface via the actionPerformed() method. The main window's actionPerformed() method determines which test was selected based on the menu item name and creates an AsyncCommandExecuter using the appropriate CommandType enumeration value. This AsyncCommandExecuter is then added to the CommandQueue where it will be executed asynchronously, and the actionPerformed() method returns, allowing the GUI to remain responsive to events (such as the update of its text area where it shows test progress). The CommandQueue calls the AsyncCommandExecuter execute() method which calls the proper TestActivationListener method based on the command type specified during construction of the AsyncCommandExecuter. The TestRunner, which implements the TestActivationListener, performs processing specific to the test that was activated. In the diagram, the pollNow test is shown, however processing for the other tests is very similar. The TestRunner first gets a connected port. The type of port and the specifics of how the connection is made are based on the settings specified in the CommSettings object. When this method returns, either a direct RS232 port is available for use or a TCP/IP port is ready. If any error occurred while connecting, the test result listener is notified and the test ends. Otherwise, an NTCIP protocol handler is created and the appropriate method is called to execute the desired test. If the test succeeds an appropriate message (or messages) are passed to the TestResultListener via the writeln() method. In the case of a pan left test, the current status is also sent to the TestResultListener for display to the user. Similarly, if the test fails, one or more messages are written to the TestResultListener via the writeln() method. The TestResultListener is the main window, and its writeln method writes data to its text area which allows the user to track test progress.



**Figure 7-27. CameraNTCIPComplianceTester:PanLeft (Sequence Diagram)**

## **7.5 Device Utility**

### **7.5.1 Class Diagrams**

#### **7.5.1.1 DeviceUtility (Class Diagram)**

This class diagram shows utility classes that are useful for tasks in performing device control.

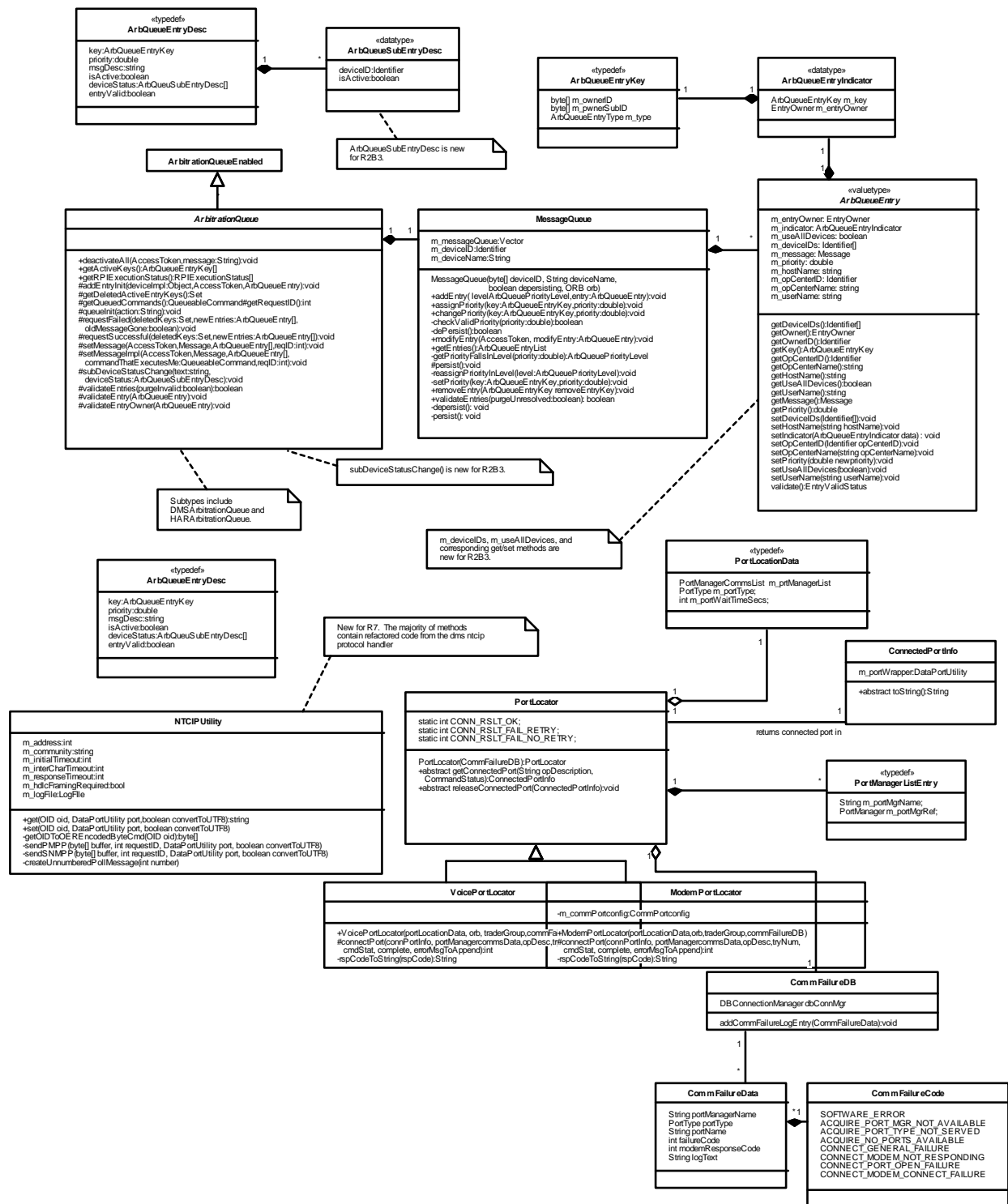


Figure 7-28. DeviceUtility (Class Diagram)



#### **7.5.1.1.1 ArbitrationQueue (Class)**

This is an abstract implementation of a generic device arbitration queue. It basically implements of the ArbitrationQueue CORBA interface (shown as ArbitrationQueueEnabled in this design). However, the official implmenters of ArbitrationQueue (ArbitrationQueueEnabled) interface are the devices themselves, CHART2DMSImpl and HARImpl. All ArbitrationQueue types of operations are delegated to an instance of this ArbitrationQueue class (one per physical device, i.e., one per instance of a device Impl class). There are device-specific concrete extensions of ArbitrationQueue for DMS and HAR, namely, DMSArbitrationQueue and HARArbitrationQueue. These provide device-specific variation.

#### **7.5.1.1.2 ArbitrationQueueEnabled (Interface)**

(This interface, defined in the design in SystemInterfaces/DeviceManagement, is called ArbitrationQueue in the code, but cannot be called ArbitrationQueue in the design because there is also an ArbitrationQueue abstract class.) An ArbitrationQueue is a queue that arbitrates the usage of a device. The evaluation of the queue determines which message(s) should be on the device, based upon the priority of the queue entries. When entries are added to the queue, they are assigned a priority level based on the type of traffic event with which they are associated, and also upon the current contents of the queue. The priority of the queue entries can be modified after they are added to the queue. The queue is evaluated when the device is online and queue entries are added or removed, when an entry's priority is modified, or when the device is put online.

#### **7.5.1.1.3 ArbQueueEntry (Class)**

This class is used for an entry on the arbitration queue, for a single message, and for a single traffic event. (It is possible, in the case of HARNotifierArbQueueEntry objects, that certain ArbQueueEntries can be on behalf of multiple TrafficEvents. In such cases, one TrafficEvent among all those involved is picked to be the responsible TrafficEvent stored in m\_indicator, the ArbQueueEntryIndicator for the entry.)

#### **7.5.1.1.4 ArbQueueEntryDesc (Class)**

This structure is used to provide a description of an entry on the arbitration queue.

#### **7.5.1.1.5 ArbQueueEntryIndicator (Class)**

The ArbQueueEntryIndicator contains data necessary to specify a unique ArbQueueEntry object; in addition, it contains a reference to the TrafficEvent which is responsible for the entry.

#### **7.5.1.1.6 ArbQueueEntryKey (Class)**

This class contains the Traffic Event ID and RPI ID and is used to identify a specific ArbQueueEntry. In some cases (e.g., for HARNotifierArbQueueEntry objects), the RPI ID is the string representing a null Identifier.

#### **7.5.1.1.7 ArbQueueSubEntryDesc (Class)**

This structure holds ArbQueueEntry "device-level detail for one "sub-device (such as a constituent HAR within a SyncHAR). It holds the ID of the device and an indication as to whether the entry is active for this particular subdevice. An ArbQueueEntry for a conglomerate device (such as a SyncHAR) will contain a list of these structures, one for each constituent HAR the entry is destined for.

#### **7.5.1.1.8 CommFailureCode (Class)**

This class defines static values to be used to specify the type of comm failure in a CommFailureData object.

#### **7.5.1.1.9 CommFailureData (Class)**

This class holds data to be passed to the CommFailureDB class to be logged in the Comm failure log in the database.

#### **7.5.1.1.10 CommFailureDB (Class)**

This class is a utility used to log an entry in the Comm Failure log table in the database. This table is used to log details about any comm failure that occurs in the system.

#### **7.5.1.1.11 ConnectedPortInfo (Class)**

This class holds data pertaining to a port that was acquired and connected via the PortLocator.

#### **7.5.1.1.12 MessageQueue (Class)**

This class represents a message queue object. It will provide the ability to add, remove, and reprioritize traffic event entries in a prioritized list.

#### **7.5.1.1.13 ModemPortLocator (Class)**

This class provides an implementation of the PortLocator's abstract connectPort() method that can connect a ModemPort that has been acquired by the PortLocator base class. This derived class logs information in the comm failure database table relating to connection problems that may occur.

#### **7.5.1.1.14 NTCIPUtility (Class)**

This class contains common utility methods for NTCIP device communications. A large portion of this class is methods refactored from the NTCIP DMS protocol handler implementation.

#### **7.5.1.1.15 PortLocationData (Class)**

This class contains configuration data that specifies the communication server(s) to use to communicate with a device.

m\_commsData - One or more objects identifying the communications server (PortManager) to use to communicate with the device, in order of preference.

m\_portType - The type of port to use to communicate with the device (ISDN modem, POTS modem, direct, etc.)

m\_portWaitTimeSecs - The maximum number of seconds to wait when attempting to acquire a port from a port manager.

#### **7.5.1.1.16 PortLocator (Class)**

The PortLocator is a utility class that helps one to connect to the port used by the device. The actual implementation of the operations is done by the derived classes depending on what protocol is used for communication.

#### **7.5.1.1.17 PortManagerListEntry (Class)**

This class is used by the PortLocator to map object identifiers to object references for PortManager objects.

#### **7.5.1.1.18 VoicePortLocator (Class)**

This class provides an implementation of the PortLocator's abstract connectPort() method that can connect a VoicePort that has been acquired by the PortLocator base class. This derived class logs information in the comm failure database table relating to connection problems that may occur. Since this is a telephony port which is much simpler to connect than, say, a ModemPort, there will be considerably fewer types of errors which can occur and thus be detected and reported.

## 7.5.1.2 PortLocatorClasses (Class Diagram)

This class diagram shows utility classes that can be used to get a free port.

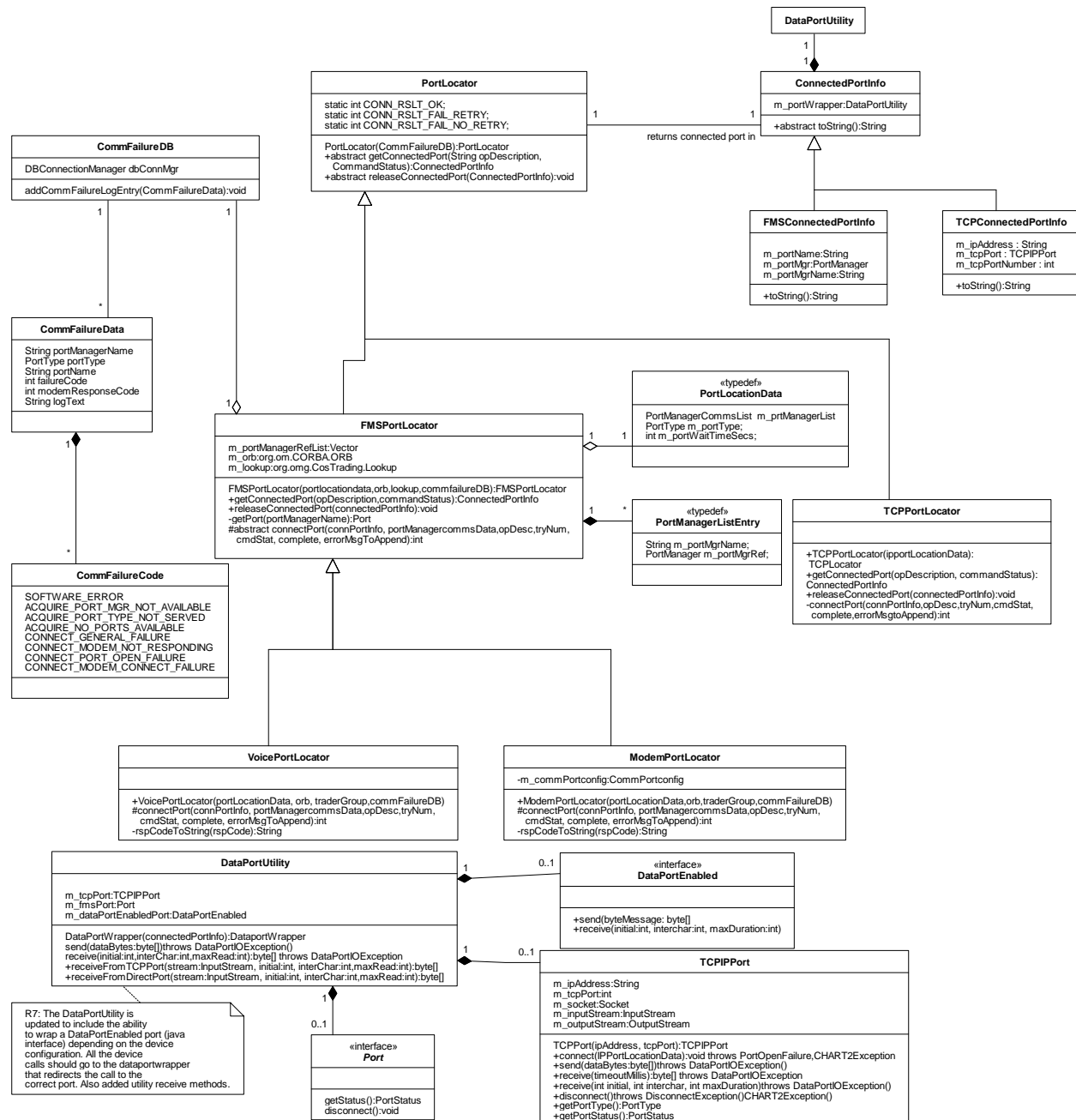


Figure 7-29. PortLocatorClasses (Class Diagram)

#### **7.5.1.2.1 CameraControlDevice (Class)**

The CameraControlDevice interface is implemented by classes which provides communications for access to control functions for a video camera. This includes encoders, command processors, and direct COM ports.

#### **7.5.1.2.2 CommFailureCode (Class)**

This class defines static values to be used to specify the type of comm failure in a CommFailureData object.

#### **7.5.1.2.3 CommFailureData (Class)**

This class holds data to be passed to the CommFailureDB class to be logged in the Comm failure log in the database.

#### **7.5.1.2.4 CommFailureDB (Class)**

This class is a utility used to log an entry in the Comm Failure log table in the database. This table is used to log details about any comm failure that occurs in the system.

#### **7.5.1.2.5 ConnectedPortInfo (Class)**

This class holds data pertaining to a port that was acquired and connected via the PortLocator.

#### **7.5.1.2.6 DataPortEnabled (Class)**

This interface is implemented by device specific communications classes. This interface provides an extra layer to remove dependencies on device specific packages.

#### **7.5.1.2.7 DataPortUtility (Class)**

This class is a wrapper used to hide the underlying port being used to communicate (tcp/ip, FMS, or CameraControlDevice port).

#### **7.5.1.2.8 FMSCONNECTEDPortInfo (Class)**

This structure defines the data used to store and exchange information about a connected port. It is returned from the PortLocator's getConnectedPort() method and is passed back into the PortLocator's release() method when it is time to release the port.

#### **7.5.1.2.9 FMSPortLocator (Class)**

The FMSPortLocator is a utility class that helps one to utilize the fault tolerance provided by the deployment of many PortManagers. The FMSPortLocator is initialized by specifying a preferred PortManager and optionally one or more alternate PortManagers using a PortLocationData object.

When asked to get a connected port, the PortLocator first attempts to acquire a port from

the preferred PortManager and then calls its abstract connectPort() method (implemented by derived classes) to attempt to connect to the port. If a failure occurs, the FMSPortLocator retries the sequence using the next PortManager in the list. The list may contain the same port manager multiple times to have retries occur on the same port manager prior to moving to another. In the event that the FMSPortLocator will perform a retry on the same port manager, it holds the previously acquired port while performing the retry to avoid having the port manager return the same port during the retry. When a different port is acquired during a retry on the same port manager, the port is released (prior to connecting the 2nd port).

#### **7.5.1.2.10 ModemPortLocator (Class)**

This class provides an implementation of the PortLocator's abstract connectPort() method that can connect a ModemPort that has been acquired by the PortLocator base class. This derived class logs information in the comm failure database table relating to connection problems that may occur.

#### **7.5.1.2.11 Port (Class)**

A Port is an object that models a physical communications resource. Derived interfaces specify various types of ports. All ports must be able to supply their status when requested.

#### **7.5.1.2.12 PortLocationData (Class)**

This class contains configuration data that specifies the communication server(s) to use to communicate with a device.

m\_commsData - One or more objects identifying the communications server (PortManager) to use to communicate with the device, in order of preference.

m\_portType - The type of port to use to communicate with the device (ISDN modem, POTS modem, direct, etc.)

m\_portWaitTimeSecs - The maximum number of seconds to wait when attempting to acquire a port from a port manager.

#### **7.5.1.2.13 PortLocator (Class)**

The PortLocator is a utility class that helps one to connect to the port used by the device. The actual implementation of the operations is done by the derived classes depending on what protocol is used for communication.

#### **7.5.1.2.14 PortManagerListEntry (Class)**

This class is used by the PortLocator to map object identifiers to object references for PortManager objects.

#### **7.5.1.2.15 TCPIPPort (Class)**

This class provides access to a TCP/IP port for device communications.

#### **7.5.1.2.16 TCPConnectedPortInfo (Class)**

This structure defines the data used to store and exchange information about a connected port. It is returned from the PortLocator's getConnectedPort() method and is passed back into the PortLocator's release() method when it is time to release the port.

#### **7.5.1.2.17 TCPPortLocator (Class)**

TCPPortLocator is a utility class that helps to establish and manage connection to a tcpip port.

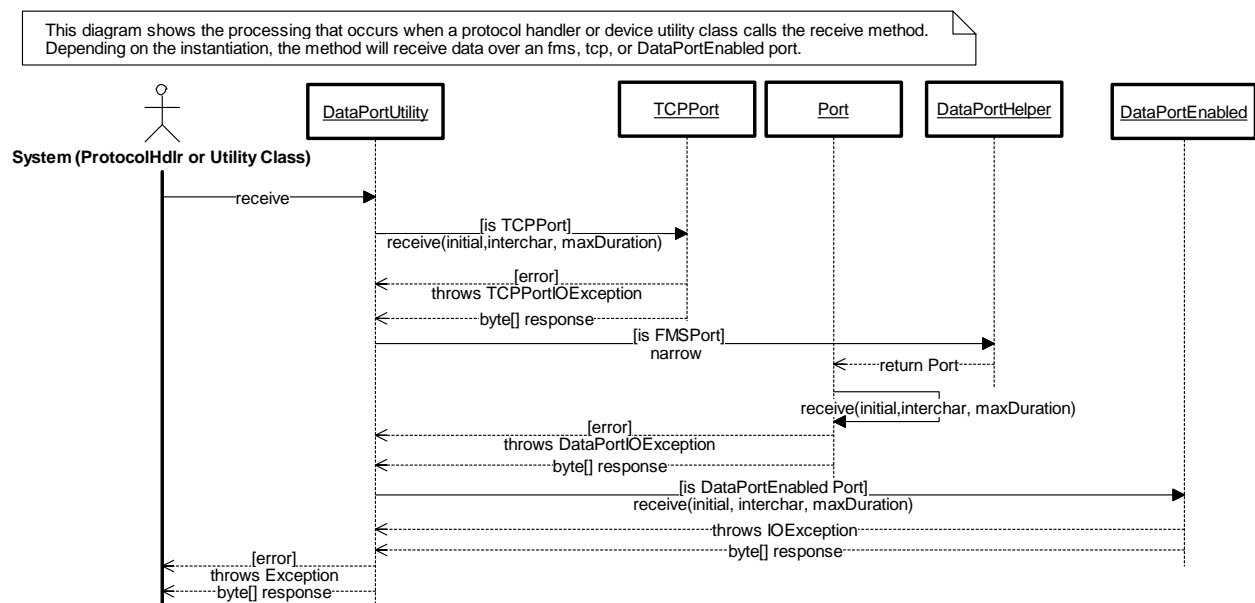
#### **7.5.1.2.18 VoicePortLocator (Class)**

This class provides an implementation of the PortLocator's abstract connectPort() method that can connect a VoicePort that has been acquired by the PortLocator base class. This derived class logs information in the comm failure database table relating to connection problems that may occur. Since this is a telephony port which is much simpler to connect than, say, a ModemPort, there will be considerably fewer types of errors which can occur and thus be detected and reported.

## 7.5.2 Sequence Diagrams

### 7.5.2.1 DataPortUtility:receive (Sequence Diagram)

This diagram shows the processing that occurs when a protocol handler or device utility class calls the receive method. Depending on the instantiation, the method will receive data over an fms, tcp, or CameraControlDevice port.

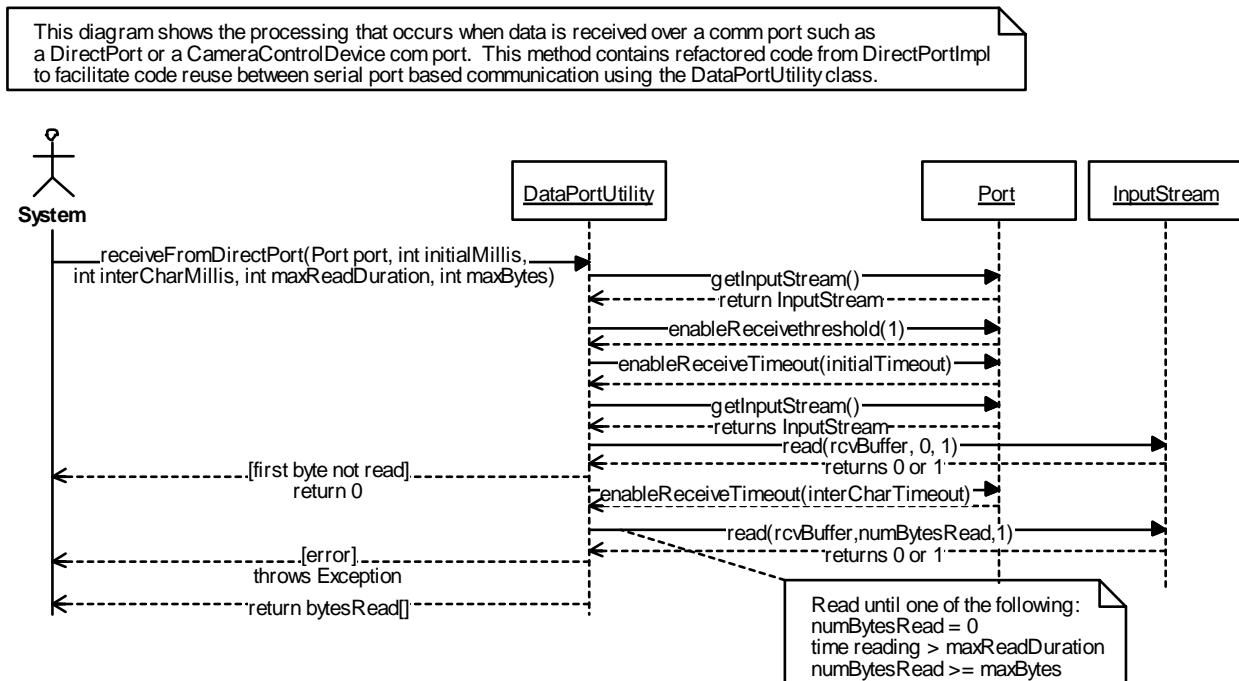


**Figure 7-30. DataPortUtility:receive (Sequence Diagram)**



### 7.5.2.2 DataPortUtility:receiveFromDirectPort (Sequence Diagram)

This diagram shows the processing that occurs when data is received over a comm port such as a DirectPort or a CameraControlDevice com port. This method contains refactored code from DirectPortImpl to facilitate code reuse between DataPortUtility and DirectPortImpl.



**Figure 7-31. DataPortUtility:receiveFromDirectPort (Sequence Diagram)**

### 7.5.2.3 DataPortUtility:receiveFromTCPPort (Sequence Diagram)

This diagram shows the processing that occurs when data is received over a comm port such as an Encoder or a TCP port. This method contains refactored code from TCPPort to facilitate code reuse between tcp based port communications using the DataPortUtility class.

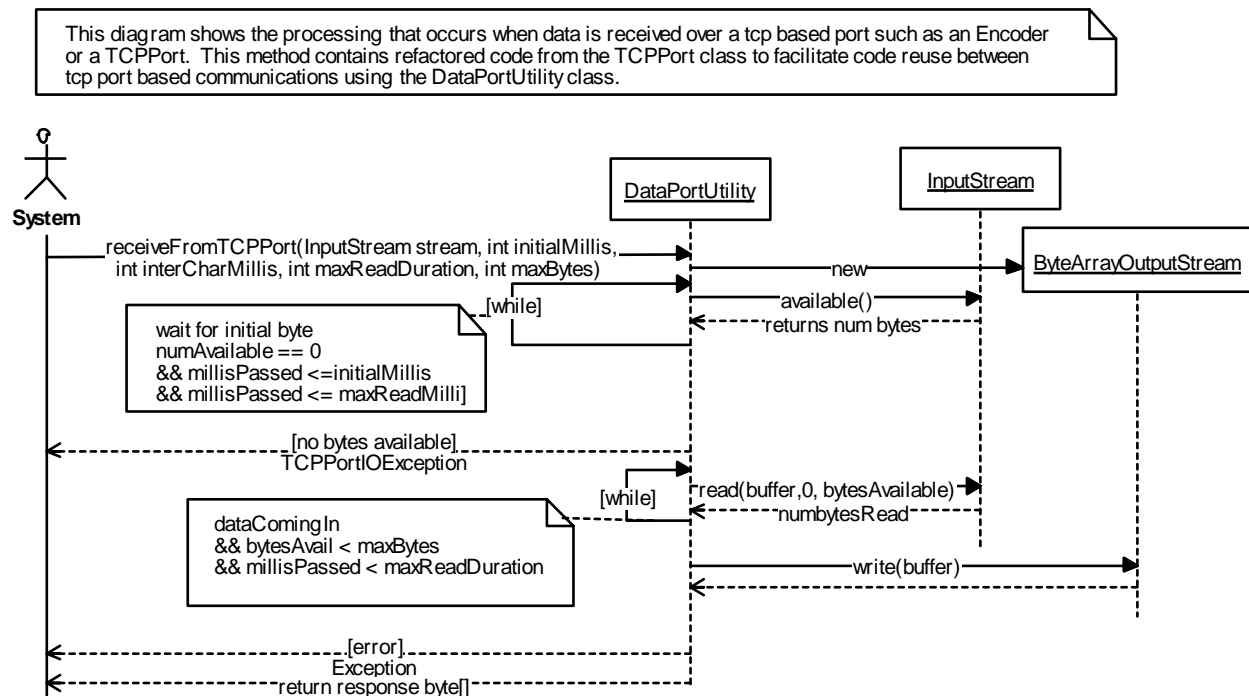


Figure 7-32. DataPortUtility:receiveFromTCPPort (Sequence Diagram)

#### 7.5.2.4 DataPortUtility:send (Sequence Diagram)

This diagram shows the processing that occurs when a protocol handler or device utility class calls the send method. Depending on the instantiation, the method will send data over an fms, tcp, or encoder port.

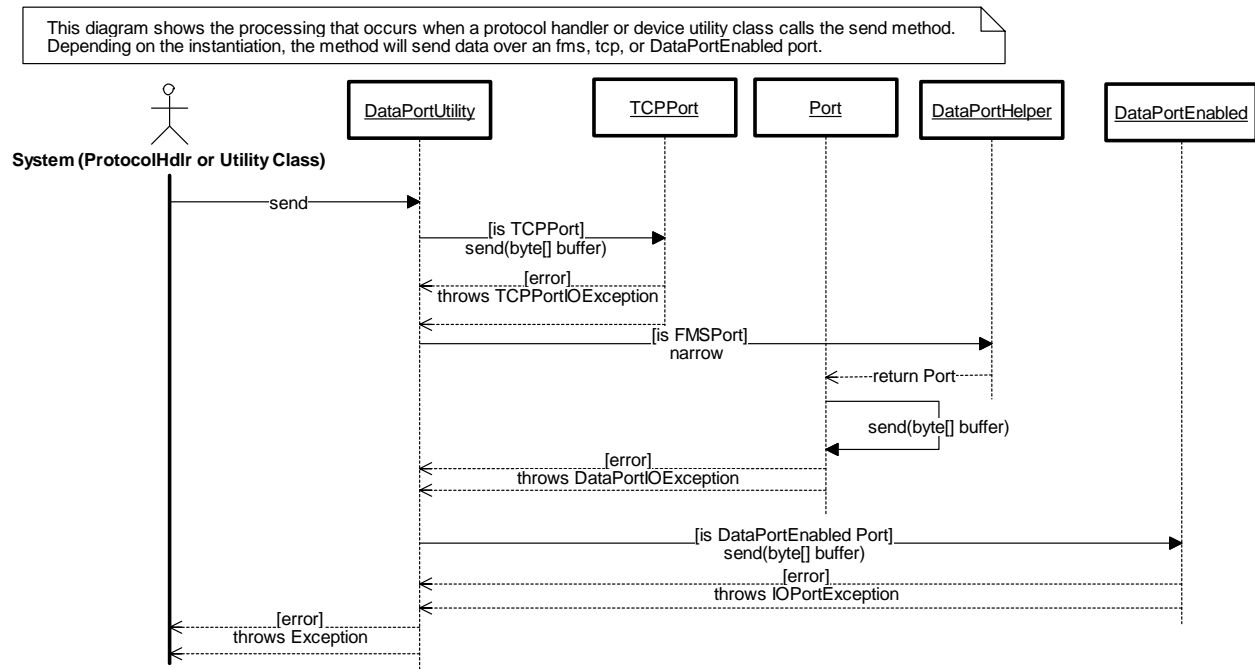
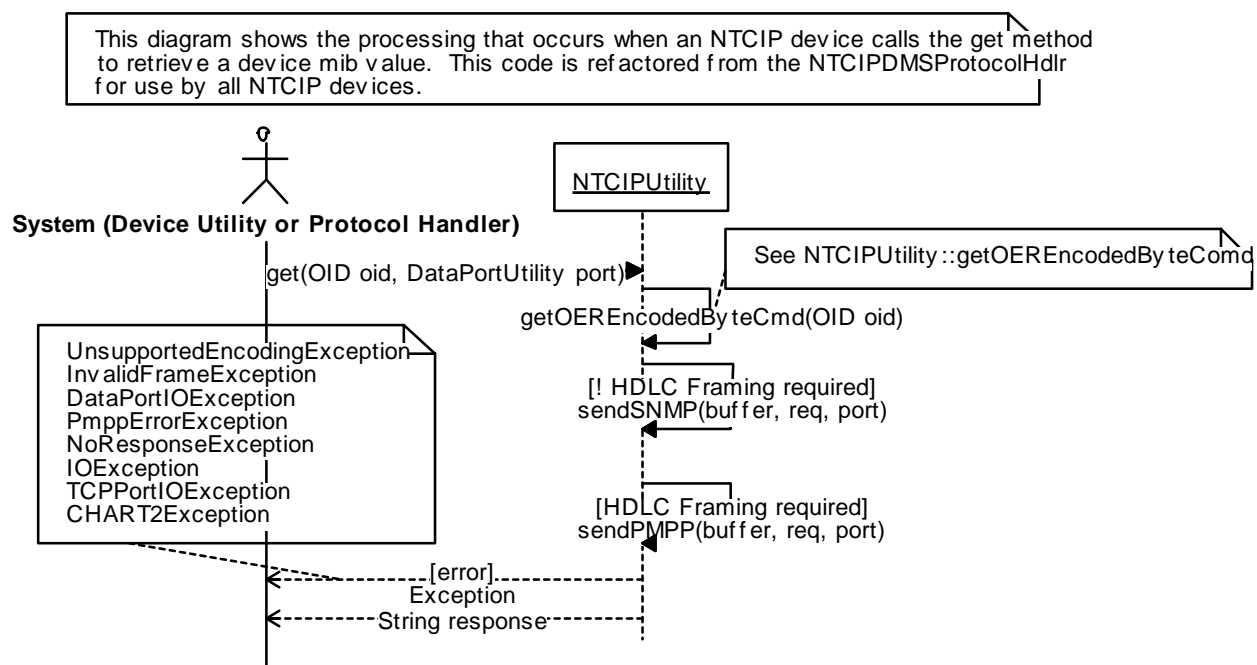


Figure 7-33 DataPortUtility:send (Sequence Diagram)

### 7.5.2.5 NTCIPUtility:get (Sequence Diagram)

This diagram shows the processing that occurs when an NTCIP device calls the get method to retrieve a device mib value. This code is refactored from the NTCIPDMSProtocolHdlr for use by all NTCIP devices.



**Figure 7-34. NTCIPUtility:get (Sequence Diagram)**

### 7.5.2.6 NTCIPUtility:getOEREncodedByteCommand (Sequence Diagram)

This method shows the processing that occurs when generating the raw ntcip byte array command. The command is either an SnmpPdu or Pmpp request. This method contains refactored code from the NTCIP DMS implementation.

This method shows the processing that occurs when generating the raw ntcip byte array command. The command is either an SnmpPdu or Pmpp request. This method contains refactored code from the NTCIP DMS implementation.

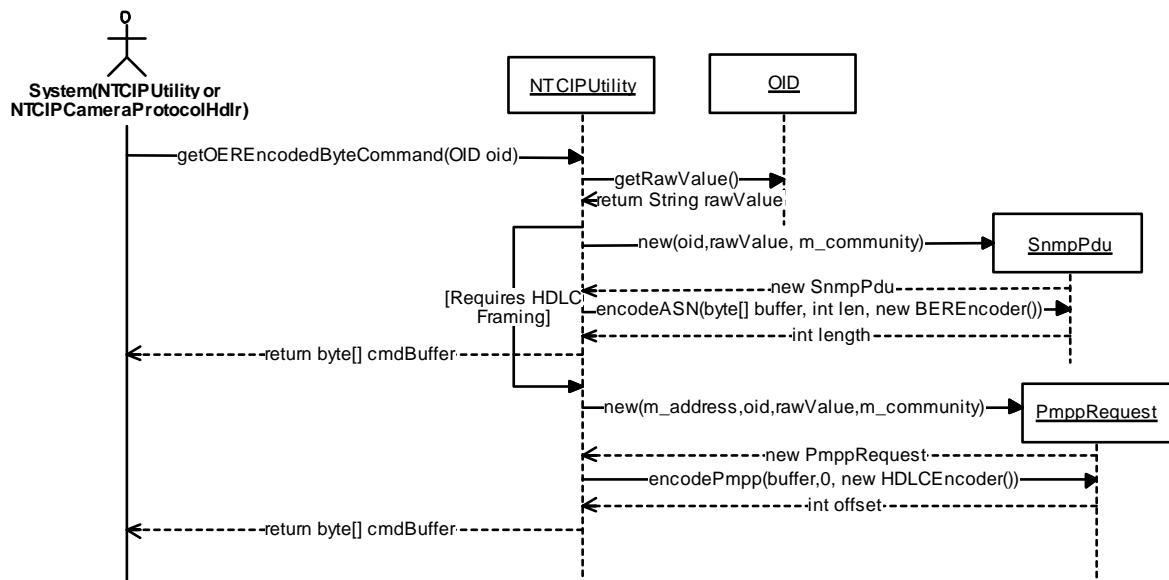


Figure 7-35. NTCIPUtility:getOEREncodedByteCommand (Sequence Diagram)

### 7.5.2.7 NTCIPUtility:set (Sequence Diagram)

This diagram shows the processing that occurs when an NTCIP device calls the set method to update a device mib value or issue a command. This code is refactored from the NTCIPDMSProtocolHdlr for use by all NTCIP devices.

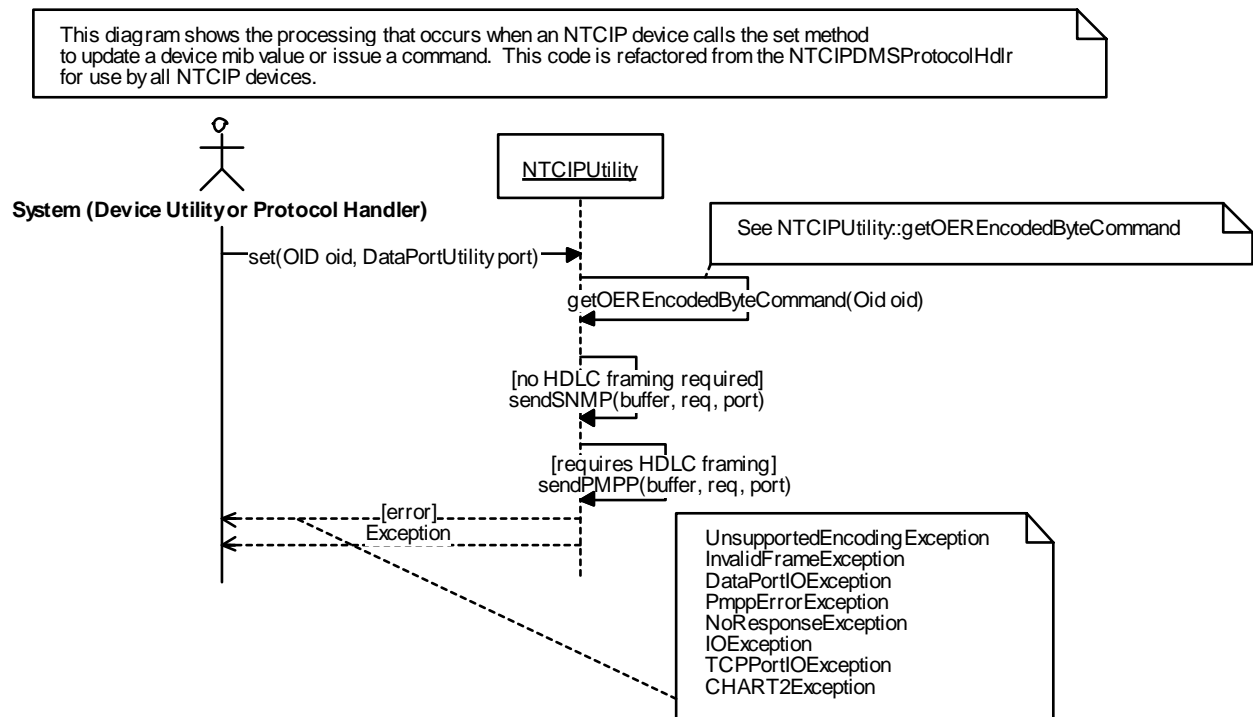


Figure 7-36. NTCIPUtility:set (Sequence Diagram)

## **8 Use Cases – SCAN Weather Integration**

---

The use case diagrams depict new functionality for the SCAN Weather Integration feature and also identify existing features that will be enhanced. The use case diagrams for the SCAN Weather Integration feature exist in the Tau design tool in the Release7 area. The sections below indicate the title of the use case diagrams that apply to this feature.

## 8.1 R7HighLevel (Use Case Diagram)

This diagram shows the high level use cases for features added or modified as part of R7.

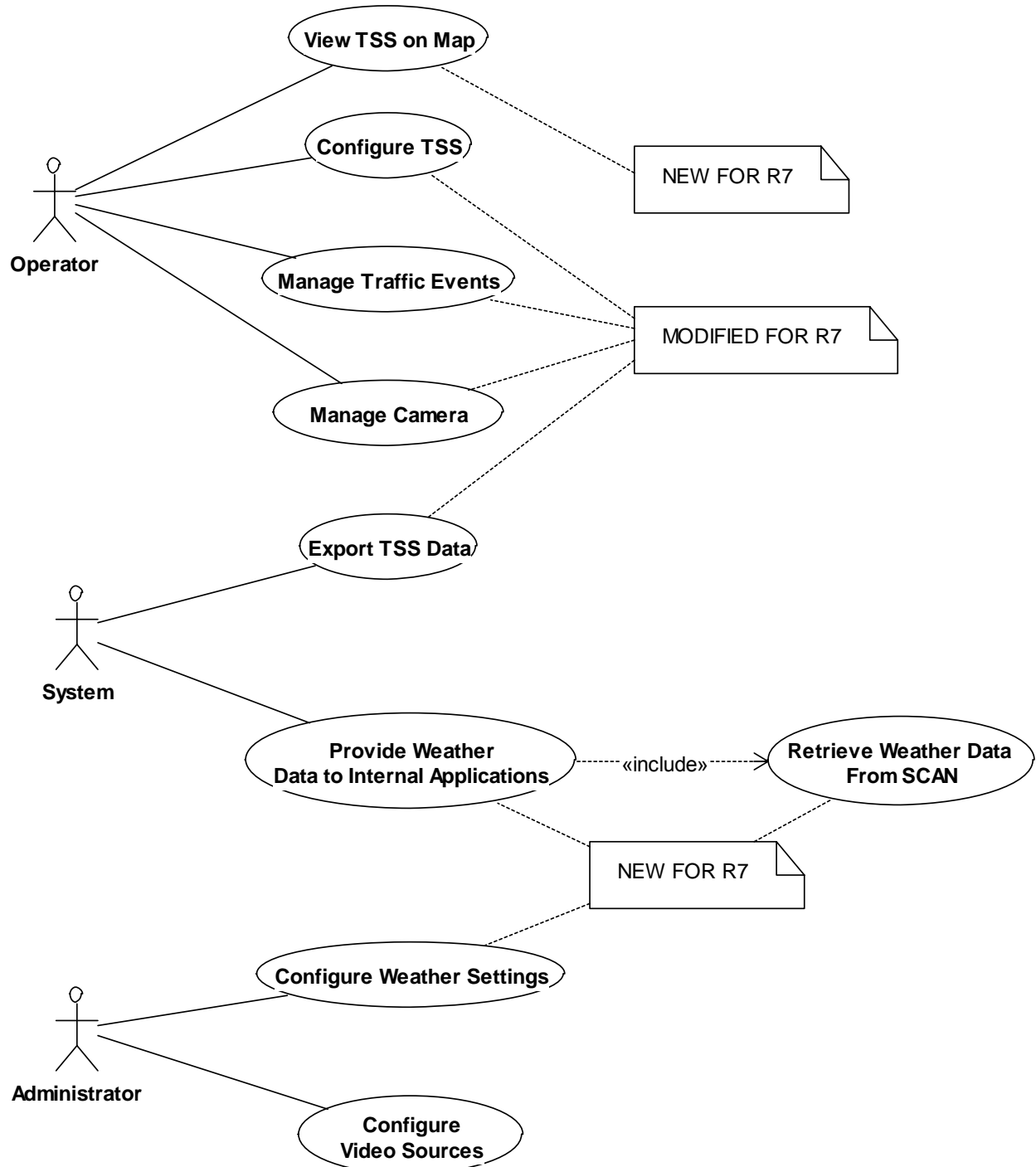


Figure 8-1. R7HighLevel (Use Case Diagram)



### **8.1.1 Configure TSS (Use Case)**

New in R7, a user can edit the map display options for a TSS. A user can specify a bearing for the TSS that is used to orient the TSS zone groups when they are displayed on the maps. A user can specify whether a zone group should be displayed on maps. If a user specifies that a zone group should be displayed on the maps, the zone group can be configured to display either in the direction of the TSS bearing or in the opposite direction (180 degrees opposed) of the TSS bearing. A user can specify the order in which zone groups should appear on the maps. Zone groups with lower display order values will appear on the map closer to the TSS lat/lon position.

### **8.1.2 Configure Video Sources (Use Case)**

The system allows an administrator with the Configure Camera right to configure video sources in the CHART system. Video Sources include generic unspecified video sources, "No Video Available" sources, fixed cameras, and controllable cameras including COHU, Vicon, and NTCIP cameras. The system allows an administrator to configure multiple video sending devices for the video source.

### **8.1.3 Configure Weather Settings (Use Case)**

The administrator will be able to configure the maximum distance from a roadway traffic event that a weather station can be to be included in the pre-selection of the road surface condition for a traffic event. (This is also referred to as a "cutoff radius"). The administrator will be able to configure the maximum age that a weather station data can have and still be included in the pre-selection of the road surface condition for a traffic event.

### **8.1.4 Export TSS Data (Use Case)**

The system shall provide detector data to external systems. The system shall enforce granular, organization based user rights to allow the level of detail provided for a detector to be controlled. Two user rights (View Detailed VSO and View Summary VSO) will be used to determine if a detector's detailed volume, speed, and occupancy (VSO) data is exported, only a speed range, or no VSO data. When VSO data is provided for a detector, it will include the data for zone groups and for each zone within the group. New for R7, CHART will export map display options for each detector including: bearing, zone group display direction, and zone group display order. The detector data will be provided using the TMDD standard, with CHART extensions as needed. External systems can obtain an inventory and status of all CHART system detectors, or the ones that have changed in a certain lookback time period. They can obtain updates to the detector data (including the status) periodically with on-demand request or by subscribing to receive updates at a specified web service URL.

### **8.1.5 Manage Camera (Use Case)**

An operator with the correct functional rights may perform basic operations on a camera. Please refer to the Manage Camera Use Case diagram for more detailed information.

### **8.1.6 Manage Traffic Events (Use Case)**

This diagram models the actions that an operator may take that relate to traffic events. This includes responding to traffic events using field devices. New in R7, the system will pre-select the road surface condition based on data from the nearest weather station with recent data when the traffic event is opened. The operator can view the weather station that was used to make the automatic selection. The weather station data will be logged to the event history log when the event is opened and again when it is closed. Just as in previous releases, the user may manually select the road surface condition while the event is open.

### **8.1.7 Provide Weather Data to Internal Applications (Use Case)**

The system will provide an interface to be used by internal applications to obtain weather related data based on location.

### **8.1.8 Retrieve Weather Data From SCAN (Use Case)**

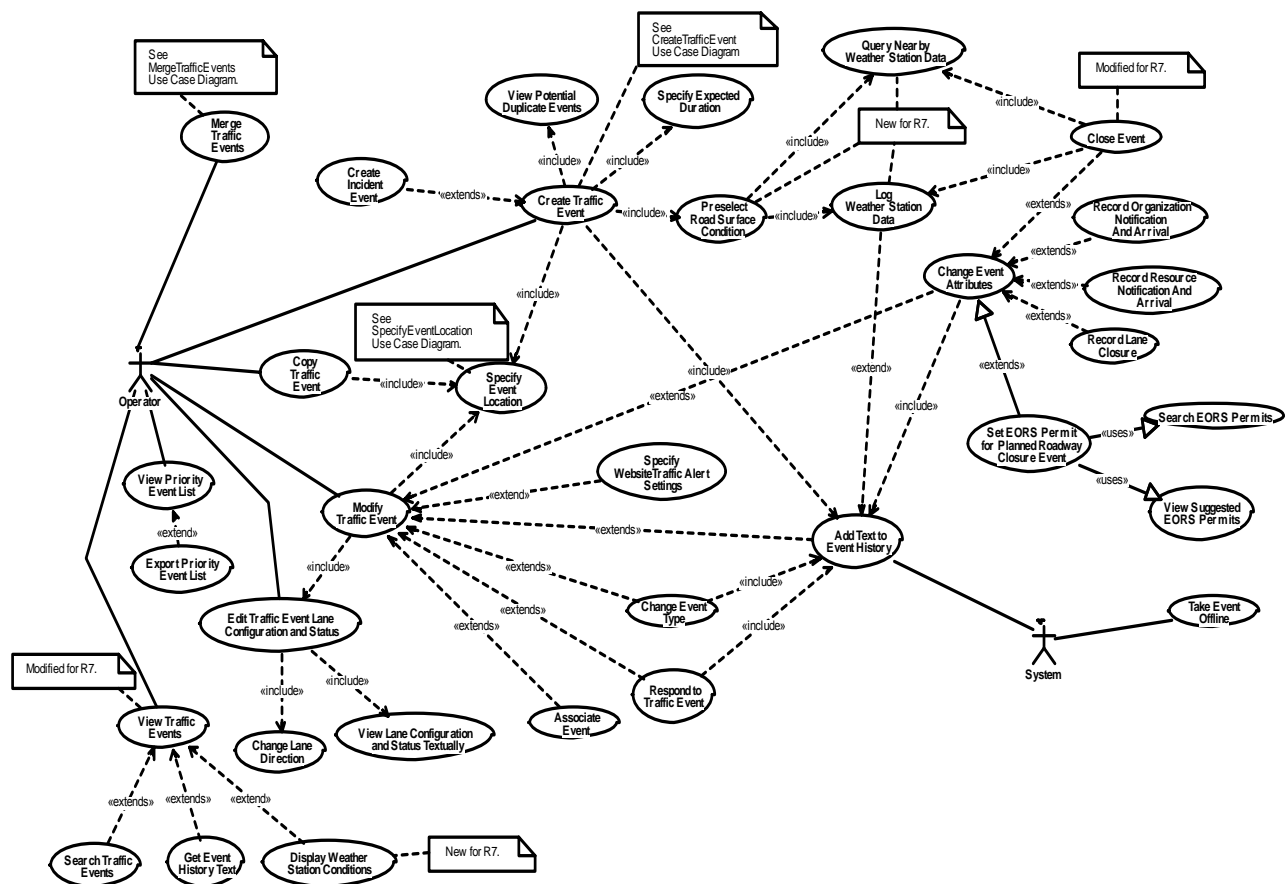
The system will retrieve weather related data from SCAN for the purpose of making it available to internal applications.

### **8.1.9 View TSS on Map (Use Case)**

The home page map shall include map layers to allow the user to view TSSs. TSS layers are shown on separate overlay layers with a separate layer for CHART TSSs and separate layers for TSSs from each external agency. The TSS layers should be below all other device layers and above the Exits/Mileposts layer. Any TSS that has a defined point location can be viewed on the home page map by clicking a link on the details page for the device.

## 8.2 ManageTrafficEvents (Use Case Diagram)

This diagram models the actions that an operator may take that relate to traffic events.



### Figure 8-2. ManageTrafficEvents (Use Case Diagram)

### 8.2.1 Add Text to Event History (Use Case)

An operator with the proper functional rights may add text to a traffic event's event history.

### 8.2.2 Associate Event (Use Case)

An operator with the proper functional rights may associate related traffic events.

### 8.2.3 Change Event Attributes (Use Case)

An operator with the proper functional rights may edit traffic event information after the event has been created. This includes closing the event, adding text to the event history, recording lane closures, and recording organization and resource arrivals. The user can also

specify the road surface conditions for a roadway event, including: wet, dry, ice/snow, wet (chemical), and unspecified.

#### **8.2.4 Change Event Type (Use Case)**

An operator with the proper functional rights may change the traffic event type.

#### **8.2.5 Change Lane Direction (Use Case)**

The lane configuration editor shall allow the user to change the traffic flow direction for a particular lane. This includes setting a lane to be multi-directional to indicate alternating use of a single lane. The editor shall allow the user to use hot keys to set the traffic flow direction. The ability to use "multi-directional" can be disabled as part of the initialization of the lane configuration editor.

#### **8.2.6 Close Event (Use Case)**

An operator with the proper functional rights may close an event. The system will query the weather station and sensor data from the nearest weather station with recent data when a traffic event is closed, if geographic coordinates are specified for the traffic event and there is a weather station within a configurable cutoff radius from that location, and will add an entry to the event history log. If the event is in the priority event list and the priority event list is in automatic mode, the system will remove the event from the priority event list when the event is closed. The system will prevent the user from closing an event if a source has not been entered for the event or if the event is an incident or planned closure and the user has not specified a lat/long for the event. The system will remind the user to fill in the queue length field and vehicles involved (for incidents) prior to closing the event (but will not prevent them from closing the event if either is not filled in, for a zero queue length and no vehicles involved are sometimes valid).

#### **8.2.7 Copy Traffic Event (Use Case)**

The user with the correct functional rights will be able to create a copy of an existing traffic event.

#### **8.2.8 Create Incident Event (Use Case)**

An operator with the proper functional rights may create a new incident event.

#### **8.2.9 Create Traffic Event (Use Case)**

The user with the correct functional rights may add a new traffic event. When creating a traffic event, the system will show the user a list of existing traffic events that may be duplicates of the new event being created based on the user's selections for the new event's location. External and pending events do not appear as possible duplicate events.

#### **8.2.10 Display Weather Station Conditions (Use Case)**

When viewing a traffic event, the system will display the weather station data that was captured at the time when the system preselected the road condition (i.e., when the event was opened). The station-level data will include the name of the weather system, a description of the location of the station, the distance from the traffic event to the station, the road surface condition that was selected for the traffic event, wind speed, wind direction, visibility, air temperature, precipitation type, precipitation intensity, and time that the weather data was obtained. It will also display per-sensor data including: a description of the location of the sensor, the road surface temperature, and the road surface condition. The user will also be able to bring up the details page for the weather station within the SCAN Web user interface.

#### **8.2.11 Edit Traffic Event Lane Configuration and Status (Use Case)**

An operator with the manage traffic events user right may edit the lane status of a traffic event, including changing direction for a particular lane. This only applies to open Planned Roadway Closures, Incidents, and Special Events.

#### **8.2.12 Export Priority Event List (Use Case)**

A user viewing the priority event list can export the displayed data as a text file. This feature allows the user to easily include information about the priority events in reports.

#### **8.2.13 Get Event History Text (Use Case)**

An operator with the correct functional rights may view the text entries that have been added to an event.

#### **8.2.14 Log Weather Station Data (Use Case)**

The system will log the weather station data at the opening and closing of a traffic event. The station-level data will include the name of the weather system, a description of the location of the station, the distance from the traffic event to the station, the road surface condition that was selected for the traffic event, wind speed, wind direction, visibility, air temperature, precipitation type, precipitation intensity, and time that the weather data was obtained. It will also display per-sensor data including: a description of the location of the sensor, the road surface temperature, and the road surface condition.

#### **8.2.15 Merge Traffic Events (Use Case)**

This use case represents the merge traffic event operation. A user with manage traffic event right merges the data of two traffic events. See MergeTraffic Events use case diagram.

### **8.2.16 Modify Traffic Event (Use Case)**

An operator with the proper functional rights may edit traffic event information after the event has been created. This includes responding to the event, editing lane status, editing location, associating with another event, and specifying other event attributes such as road condition.

### **8.2.17 Operator (Actor)**

An operator is a user of the system who has been assigned a valid username/password combination and granted roles for system access.

### **8.2.18 Preselect Road Surface Condition (Use Case)**

The system will query and pre-select the road surface condition indicated by the nearest weather station with recent data when a traffic event is opened, if geographic coordinates are specified for the traffic event and there is a weather station within a configurable radius from that location. The system will record the selected station and sensor data in the event history log.

### **8.2.19 Query Nearby Weather Station Data (Use Case)**

The system will query the nearest weather station with recent data, if geographic coordinates are specified for the traffic event and there is a weather station within a specified radius from that location. The system will calculate distance as the crow flies. The system will, if possible, use the worst road surface condition from sensors matching the primary route and direction of the traffic event. If no sensors are found matching the traffic event's route and direction, the worst road surface condition from sensors matching the traffic event's route will be used. If no sensors match the traffic event's primary route, the system will use the worst road surface condition reported by any of the station's sensors. The system will only make a road surface condition selection if the selected weather station has roadway sensors with data.

### **8.2.20 Record Lane Closure (Use Case)**

The lane configuration editor shall allow the user to specify the status of each lane in the configuration as being open, closed, or unknown. A feature shall also exist to allow the user to set all lanes to open without having to set the status individually for each lane. Hot keys for setting lane status will be supported. The system will record the date/time a lane is opened or closed. When the lane configuration is initialized to include (not disable) the feature that sets the initial lane status in the main direction to unknown, then When a user makes the first change to the status of a lane in the main direction whose status was initially defaulted to "unknown", the system will set the status of all other lanes in the main direction that have a status of "unknown" to "open".

#### **8.2.21 Record Organization Notification And Arrival (Use Case)**

An operator with the proper functional rights may record the participation of various organizations in the event resolution.

#### **8.2.22 Record Resource Notification And Arrival (Use Case)**

An operator with the proper functional rights may record the participation of various resources in the event resolution.

#### **8.2.23 Respond to Traffic Event (Use Case)**

The system allows an operator to control devices in response to an event through the use of a response plan. The user may add devices to the plan, select the desired state of the devices, then activate the plan. Any of the devices used by the event response plan may be deactivated while the event is open by removing the item for that device from the plan. When the event is closed, if the response plan is active, it will be deactivated automatically.

#### **8.2.24 Search EORS Permits (Use Case)**

The system will allow the user to perform a text search on the following fields of an EORS permit: permit tracking number, start county name, end county name, permit type, route location, route type, route number, work order description, permittee name, contract number and days of week. The system will score each matching permit based on the percentage of the user entered search terms were found in these fields and will present the results in order of relevance. The user will be shown a summary of each matching permit and will be able to show/hide additional details about each permit.

#### **8.2.25 Search Traffic Events (Use Case)**

An operator with the proper functional rights may search the CHART system for traffic events.

#### **8.2.26 Set EORS Permit for Planned Roadway Closure Event (Use Case)**

A user may set the EORS permit associated with a planned roadway closure event. Doing so will set the EORS permit tracking number into the planned roadway closure event details. The system will assist the user in finding the correct EORS permit by suggesting potential matching permits as the user types a permit tracking number. In the event that the user does not see the permit they are looking for in the list of suggested permits, the system will provide a more advanced searching capability that will search on other fields of the permit (in addition to the tracking number). The user will be able to specify if the list of permits considered for suggestion or searching should be limited to only the active and queued permits, or if all permits currently available in the EORS system should be considered.

The user will be able to indicate if permit tracking numbers should be considered to match

their search terms if the permit number starts with the user entered text (only) or contains the user entered text anywhere within the permit number. Additionally the user may indicate that the permit may start with or end with (last 4 digits) the user entered text.

### **8.2.27 Specify Event Location (Use Case)**

The event location choices will be populated using data from the CHART Mapping application database.

By default, MD will be selected as the state.

If the selected state is MD, the user will be required to select a predefined MD county/region. If a route is specified, the user will first select a route type from a pick list ("I", "US", or "MD") and the route type will be used to populate the list of predefined routes. To specify a route, the user will be required to select one of the predefined routes if the state is MD. If the state is not MD, the user will be able to enter a county name / region name and route number as freeform text.

If a route number is specified, the user will be able to select intersecting roads by route number or route name, or specify the state or county milepost. Additionally the user will be able to specify whether the traffic event is at, prior, or past the intersecting feature ("at" will be selected by default).

If the state is MD, the list of intersecting route numbers and names will be populated for the user as suggestions; however, the user can still specify freeform text for an intersecting route number, route name, county milepost, and state milepost even if the state is MD.

### **8.2.28 Specify Expected Duration (Use Case)**

An operator with the proper functional rights may specify the expected duration of an event.

### **8.2.29 Specify WebsiteTraffic Alert Settings (Use Case)**

An operator with the proper functional rights may specify whether or not the traffic event warrants a "Traffic Alert" on the public CHART web site, and may optionally provide specific alert text to be associated with the Alert.

### **8.2.30 System (Actor)**

The System actor represents any software component of the CHART system. It is used to model uses of the system which are either initiated by the system on an interval basis, or are an indirect by-product of another use case that another actor has initiated.

### **8.2.31 Take Event Offline (Use Case)**

The system periodically checks for closed events and takes them offline provided that a



configured interval of time has elapsed since the event was closed.

#### **8.2.32 View Lane Configuration and Status Textually (Use Case)**

The system shall provide a read-only textual description of the specified lane configuration and status.

#### **8.2.33 View Potential Duplicate Events (Use Case)**

An operator with the correct functional rights will be presented with a list of potential duplicate traffic events based on the event location. The operator will have the option to then merge these events.

#### **8.2.34 View Priority Event List (Use Case)**

Users may view a list of priority events, as defined automatically by the system or as defined manually by a user. When in automatic mode, the system selects the events to appear in the priority events list based on attributes of the events, such as the incident type (Fatality), the HazMat flag, the types of vehicles involved, lane closures, and queue length. When in manual mode, a user selects the events to appear in the list and the order in which they appear. The mode is controlled via the system profile. The display is a text only display and can be accessed without having to log into the CHART system. A link within the CHART system provides access to it, however users may also access it directly via its specific URL.

#### **8.2.35 View Suggested EORS Permits (Use Case)**

As the user types in the search field the system will present a set of suggested permits based on the permit tracking number. The suggested permits will each display the tracking number and some summary information about the permit as well as a link/button that allows the user to associate the permit to the planned closure directly from the suggestion.

#### **8.2.36 View Traffic Events (Use Case)**

An operator with the correct functional rights may view a traffic event. The details for a traffic event will include the weather station conditions, if the road surface condition was pre-selected by the system. While viewing the details page of a roadway event with a defined geographic location, an operator may invoke the Intranet Map to view weather stations near the traffic event.

## 9 Detailed Design – SCAN Weather Integrations

---

### 9.1 Human-Machine Interface

#### Weather Integration

##### Traffic Event Details Page

When a Traffic Event is created where an operator can select Roadway Conditions (currently Incidents and Weather Events) the Roadway Conditions section of the Traffic Event Details page will now be automatically populated.

To populate the Road Surface Condition field, the system will select data from the nearest weather station if three criteria are met: 1) the weather station must be within a given radius of the event and 2) its data must be no older than a given age and 3) the weather station must have at least one roadway sensor. Both the radius and maximum data age will be configurable via System Profile settings.

If a weather station is identified that meets these criteria, the Road Surface Condition field is assigned by matching the traffic event's roadway and direction to the roadway and direction of the weather station's roadway sensors. This matching is accomplished according to the following algorithm:

- 1) Select all roadway sensors whose roadway and direction match the traffic event
- 2) If none, select all roadway sensors whose roadway matches the traffic event
- 3) If none, select all roadway sensors

Once the (possibly empty) set of sensors is collected, the traffic event's Road Surface Condition is assigned the worst surface condition in the set according to this list of increasingly worse conditions:

- 1) UNSPECIFIED
- 2) DRY
- 3) WET
- 4) CHEMICALLY WET
- 5) ICE OR SNOW

If an automatic selection is made, other relevant weather conditions from the weather station will also be displayed. The information shown represents a snapshot of the weather data from the time when the automatic selection was made.

**Roadway Conditions**

Road Surface Condition: Wet

Nearby Wx Station: (Intranet Map)

Location: I-270 @ MD-109; Distance: 2.3 mi; Surface Condition: Wet, 33F; Air Temp: 31F; Precip: Snow, Light; Vis: 0.5 mi; Wind: 15 MPH NNW; System: SCAN; 12/10/2010 14:45

(show/hide sensors) (SCAN details)

**Figure 9-1 Roadway Conditions Section in Traffic Event**

The user can override the automatically-selected Road Surface Condition by making another selection; however, the Nearby Wx Station information will still be displayed.

(Normally the automatic selection will appear as soon as the traffic event is created, but if the call to get the weather information takes longer than a configurable amount of time, this information may be filled in after event creation, in which case it would appear on the next page refresh. The amount of time will be configurable via a System Profile setting. This mechanism will prevent excessive delays while creating Incidents or Weather Service Events if there is a problem getting the data.)

If the system is able to set the Road Surface Condition according to the criteria described above, then it will record this in the traffic event history log. Similarly when a traffic event that supports Roadway Conditions is closed, a separate Roadway Conditions history log entry will be made. Because the same algorithm is applied at event closure, it is possible a different roadway sensor may be used than at event creation (e.g. if the traffic event location or weather system configuration has changed in the interim).

The ***Intranet Map*** link will allow the user to bring up the Intranet Map page to see nearby weather stations in a popup window, if the traffic event has coordinates defined and the Intranet Map is available.

The ***SCAN details*** link will allow the user to bring up the SCAN Web application to see details for the weather station in a popup window, if data from a weather station was selected and the SCAN Web application is available.

The ***show / hide sensors*** link will allow the user to show or hide details for the roadway sensors. The actual implementation will display “show” when the roadway sensor details are hidden and “hide” when they are shown. The details will be hidden initially, to avoid clutter and to avoid using vertical space on the page. When the sensor information is displayed, it will show the location description, route and direction, surface condition, and surface temperature for each sensor:

Roadway Conditions

Road Surface Condition: Wet

Nearby Wx Station:

Location: I-270 @ MD-109; Distance: 2.3 mi; Surface Condition: Wet, 33F; Air Temp: 31F; Precip: Snow, Light; Vis: 0.5 mi; Wind: 15 MPH NNW; System: SCAN; 12/10/2010 14:45

[\(Intranet Map\)](#)

[\(show/hide sensors\)](#)

[\(SCAN details\)](#)

Location	Route / Direction	Surface Condition	Surface Temp
I-270 SB Departure (0)	I-270 SOUTH	Dry	34F
I-270 SB Deck (1)	I-270 SOUTH	Dry	36F
I-270 NB Deck (2)	I-270 NORTH	Dry	35F
MD-109 NB (3)	MD 109 NORTH	Wet	33F

**Figure 9-2 Detailed Roadway Conditions**

CHART R7 Detailed Design

9-14

03/02/2011

## 9.2 System Interfaces

### 9.2.1 Class Diagrams

#### 9.2.1.1 TrafficEventManagement2 (Class Diagram)

This diagram shows more interface classes related to traffic events.

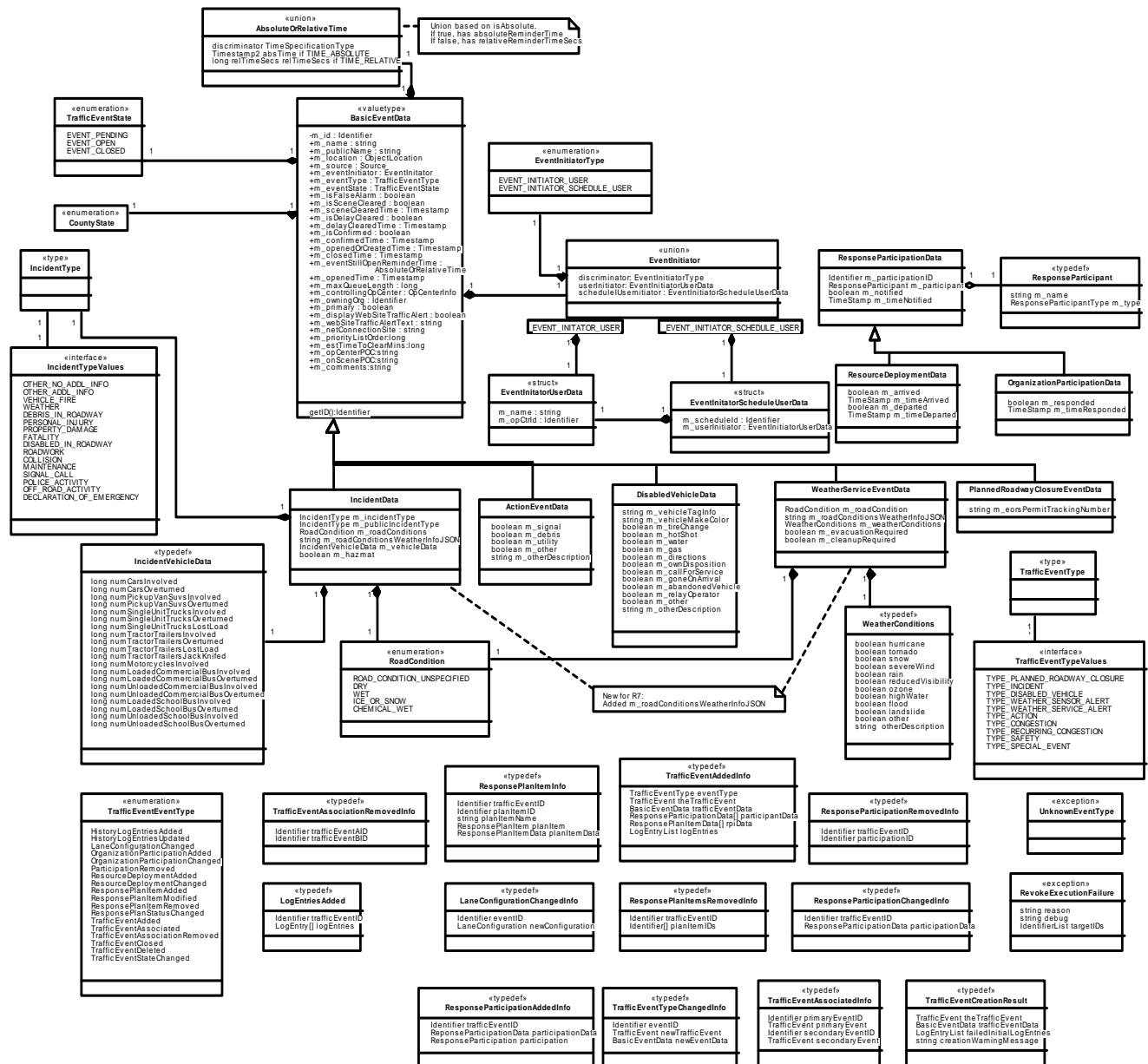


Figure 9-3. TrafficEventManagement2 (Class Diagram)

#### **9.2.1.1.1 AbsoluteOrRelativeTime (Class)**

This union stores a time, in either absolute or relative terms.

#### **9.2.1.1.2 ActionEventData (Class)**

This class represents all data specific to an Action event type traffic event.

#### **9.2.1.1.3 BasicEventData (Class)**

This class represents the data common to all traffic events. All derived data types will inherit all data shown in this class.

#### **9.2.1.1.4 CountyState (Class)**

This enumeration defines the various counties in Maryland and the states surrounding Maryland that will be used for defining the traffic event.

#### **9.2.1.1.5 DisabledVehicleData (Class)**

This class represents all data specific to a disabled vehicle traffic event.

#### **9.2.1.1.6 EventInitiatorScheduleUserData (Class)**

This structure contains data about a schedule involved in the initiation of a traffic event. It is contained within the EventInitiator union.

#### **9.2.1.1.7 EventInitiatorUserData (Class)**

This structure contains data about a user involved in the initiation of a traffic event. It is contained within the EventInitiator union.

#### **9.2.1.1.8 EventInitiator (Class)**

This union contains information about the entity or entities involved in the initiation of a traffic event. This can be the schedule, if a schedule was involved in initiating the event, and/or a user, if a user was involved in initiating the event. This union allows for possible expansion in future releases, where traffic events may be initiated by a schedule without user confirmation, or by CHART devices (traffic sensors, weather sensors, etc.) or external interfaces (RITIS, etc.) initially with, or possibly later without, user involvement.

#### **9.2.1.1.9 EventInitiatorType (Class)**

This enumeration identifies the types of initiators which can initiate traffic events. Traffic events can be initiated by a user (directly), or by a schedule (with user involvement). This enumeration, and the union in which it is a discriminator, allows for possible expansion in future releases, where traffic events may be initiated by a schedule without user

confirmation, or by CHART devices (traffic sensors, weather sensors, etc.) or external interfaces (RITIS, etc.) initially with, or possibly later without, user involvement.

#### **9.2.1.1.10 IncidentData (Class)**

This class represents data specific to an Incident type traffic event.

#### **9.2.1.1.11 IncidentType (Class)**

This typedef defines the type of the incident.

#### **9.2.1.1.12 IncidentTypeValues (Class)**

This interface lists all possible incident types.

#### **9.2.1.1.13 IncidentVehicleData (Class)**

This class represents the vehicles involved data for incidents. Its purpose is to simplify the exchange of data between GUI and server.

#### **9.2.1.1.14 LaneConfigurationChangedInfo (Class)**

This structure contains the data that is broadcast when the lane configuration of a traffic event is changed.

#### **9.2.1.1.15 LogEntriesAdded (Class)**

This structure contains the data that is broadcast when new entries are added to the event history log of a traffic event.

#### **9.2.1.1.16 OrganizationParticipationData (Class)**

This class represents the data required to describe an organization's participation in the response to a traffic event.

#### **9.2.1.1.17 PlannedRoadwayClosureEventData (Class)**

This class contains data specific to the PlannedRoadwayEvent type of traffic event.

#### **9.2.1.1.18 ResourceDeploymentData (Class)**

This class represents the data required to describe a resource's participation in the response to a traffic event.

#### **9.2.1.1.19 ResponseParticipant (Class)**

The ResponseParticipant class is a non-behavioral structure which specifies a participant in a response.

#### **9.2.1.1.20 ResponseParticipationAddedInfo (Class)**

This structure contains the data that is broadcast when a response participant is added to the response to a particular traffic event.

#### **9.2.1.1.21 ResponseParticipationChangedInfo (Class)**

This structure contains the data pushed in a CORBA event any time any type of response participation object changes state.

#### **9.2.1.1.22 ResponseParticipationData (Class)**

This class contains all data pertinent to any class that represents a response participation.

#### **9.2.1.1.23 ResponseParticipationRemovedInfo (Class)**

This structure contains the data that is broadcast when one or more response plan items are removed from a traffic event.

#### **9.2.1.1.24 ResponsePlanItemInfo (Class)**

This structure contains the data that is broadcast any time a new response plan item is added or an existing response plan item is modified.

#### **9.2.1.1.25 ResponsePlanItemsRemovedInfo (Class)**

This structure contains the data that is broadcast when one or more response plan items are removed from a traffic event.

#### **9.2.1.1.26 RevokeExecutionFailure (Class)**

This class defines a exception thrown when failed to revoke a response plan item's execution.

#### **9.2.1.1.27 RoadCondition (Class)**

This enumeration lists the possible roadway conditions at the scene of a traffic event.

#### **9.2.1.1.28 TrafficEventAddedInfo (Class)**

This structure contains the data that is broadcast when a new traffic event is added to the system.

#### **9.2.1.1.29 TrafficEventAssociatedInfo (Class)**

This structure contains the data that is broadcast when two traffic events are associated.



#### **9.2.1.1.30 TrafficEventAssociationRemovedInfo (Class)**

This structure contains the data that is broadcast when the association between two traffic events is removed.

#### **9.2.1.1.31 TrafficEventCreationResult (Class)**

This result is returned from createEvent() to indicate warning messages if the event was not created cleanly.

#### **9.2.1.1.32 TrafficEventEventType (Class)**

This enumeration defines the types of CORBA events that can be broadcast on a Traffic Event related CORBA Event channel.

#### **9.2.1.1.33 TrafficEventState (Class)**

This enumeration lists the possible states for a traffic event. The states are pending, open, and closed. A false alarmed "state" is considered a special case of "closed", so false alarmed events will have a TrafficEventState of EVENT\_STATE\_CLOSED. They will also have the m\_isFalseAlarm flag in their BasicEventData set to true to distinguish them from normally closed events.

#### **9.2.1.1.34 TrafficEventType (Class)**

This typedef defines the type of traffic event.

#### **9.2.1.1.35 TrafficEventTypeChangedInfo (Class)**

This structure contains the data that is broadcast when a traffic event changes types. The traffic event object that represented the traffic event previously is removed from the system and is replaced by the newTrafficEvent reference contained in this structure. If the consumer of this CORBA event has stored any references to the traffic event previously, those references should be replaced with this new reference.

#### **9.2.1.1.36 TrafficEventTypeValues (Class)**

This interface defines the types of traffic events that are supported by the system.

#### **9.2.1.1.37 UnknownEventType (Class)**

This class defines an exception thrown when the type of a traffic event type is not known and is not defined in TrafficEventTypeValues.

#### **9.2.1.1.38 WeatherConditions (Class)**

This structure contains all possible weather conditions. Each member should be set to true if that condition applies, false otherwise. The m\_otherDescription member will only be considered valid if the m\_other member is set to true.

#### **9.2.1.1.39 WeatherServiceEventData (Class)**

This class contains data specific to the WeatherServiceEvent type of traffic event.

## 9.3 Traffic Event Module Package

### 9.3.1 Class Diagrams

#### 9.3.1.1 TrafficEventModuleClassesR7 (Class Diagram)

This diagram show Traffic Event Module classes related to the changes in Release 7.

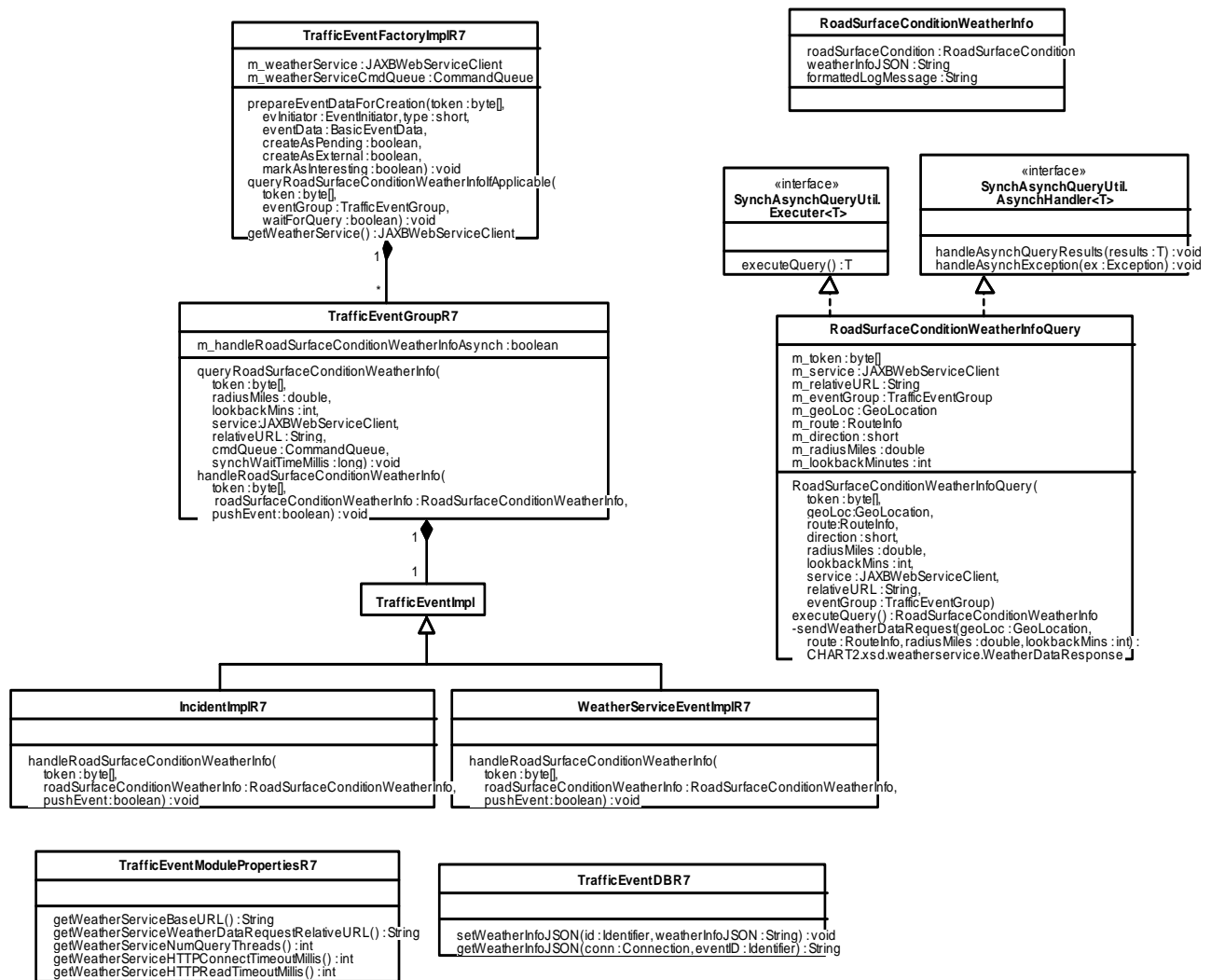


Figure 9-4. TrafficEventModuleClassesR7 (Class Diagram)

#### 9.3.1.1.1 IncidentImplR7 (Class)

This class contains IncidentImpl changes for R7. This class is the CORBA object

implementation for an Incident event.

#### **9.3.1.1.2 RoadSurfaceConditionWeatherInfo (Class)**

This class or structure is used as the internal representation (within the Traffic Event Module) of the results of querying the road surface condition weather info from the Weather Service. It contains all of the data needed by the traffic event module classes.

#### **9.3.1.1.3 RoadSurfaceConditionWeatherInfoQuery (Class)**

This class executes and handles the results of the query to the Weather web service.

#### **9.3.1.1.4 SynchAsynchQueryUtil. AsynchHandler<T> (Class)**

This interface allows an object to handle the data returned from a query asynchronously (i.e., not on the calling thread).

#### **9.3.1.1.5 SynchAsynchQueryUtil. Executer<T> (Class)**

This interface is implemented by an object that can execute a query. The type of data returned from the query is specified as a generic type. Any data needed for the query must be stored in the implementing object.

#### **9.3.1.1.6 TrafficEventDBR7 (Class)**

This class contains TrafficEventDB changes for R7. This class provides database functionality related to traffic events.

#### **9.3.1.1.7 TrafficEventFactoryImplR7 (Class)**

This class shows changes in the TrafficEventFactoryImpl class for R7. This class manages all traffic events served by the Traffic Event Service.

#### **9.3.1.1.8 TrafficEventGroupR7 (Class)**

This class shows changes in the TrafficEventGroup class for R7. This class manages a single TrafficEvent CORBA object implementation.

#### **9.3.1.1.9 TrafficEventImpl (Class)**

This class provides an implementation of the TrafficEvent interface. It contains state variables and processing that common to all traffic events.

#### **9.3.1.1.10 TrafficEventModulePropertiesR7 (Class)**

This class contains TrafficEventModuleProperties changes for R7. This class provides access to settings defined in the TrafficEventService properties file.

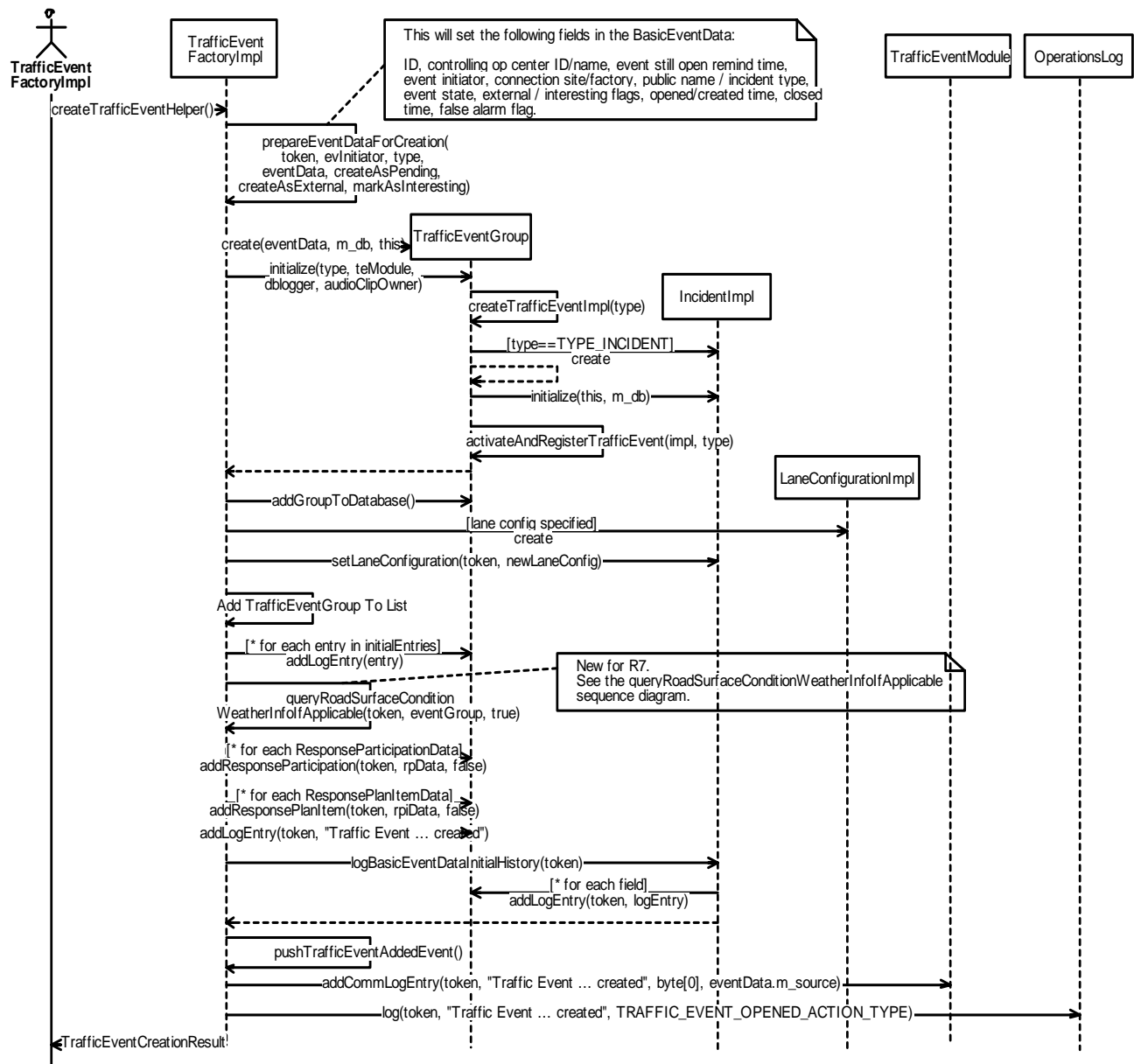
#### **9.3.1.1.11 WeatherServiceImplR7 (Class)**

This class shows WeatherServiceImpl changes for R7. This class is the CORBA object implementation for a Weather Service event.

### **9.3.2 Sequence Diagrams**

#### **9.3.2.1 TrafficEventFactoryImpl:createTrafficEventHelper (Sequence Diagram)**

This diagram shows how a traffic event is created. First, the BasicEventData that is passed in is prepared for event creation. This involves setting a number of variables, including the ID of the new event and several other fields. Next the TrafficEventGroup object is created and initialized. This creates a TrafficEventImpl object of the appropriate type and activates the object in CORBA and publishes it in the Trading Service. The TrafficEventGroup is added to the database. The lane configuration (if specified) is set. The TrafficEventGroup is added to the factory's list. The initial event history log entries are added. For R7, a call to queryRoadSurfaceConditionWeatherInfoIfApplicable() will be made to query the road surface condition weather information (this is documented in the sequence diagram of the same name). The response participation and response plan item objects are also added. Log messages are added to the even history and comm log. The TrafficEventAdded event is pushed. Finally a TrafficEventCreationResult is returned, which contains a reference to the event and any warning messages.



**Figure 9-5. TrafficEventFactoryImpl:createTrafficEventHelper (Sequence Diagram)**

### 9.3.2.2 TrafficEventFactoryImpl:getWeatherService (Sequence Diagram)

This diagram shows how the traffic event factory gets or creates a JAXBWebServiceClient object for calling the Weather Service. This object will only be created on first use - after that the existing one will be reused. A JAXBContext object is created, and the base URL for the web service is read from the properties and used to create an XMLHTTPService object. The HTTP connect and read timeout values are then set using values specified in the Traffic Event Service properties file. A JAXBWebServiceClient object is created, stored in the factory for later use, and returned to the caller.

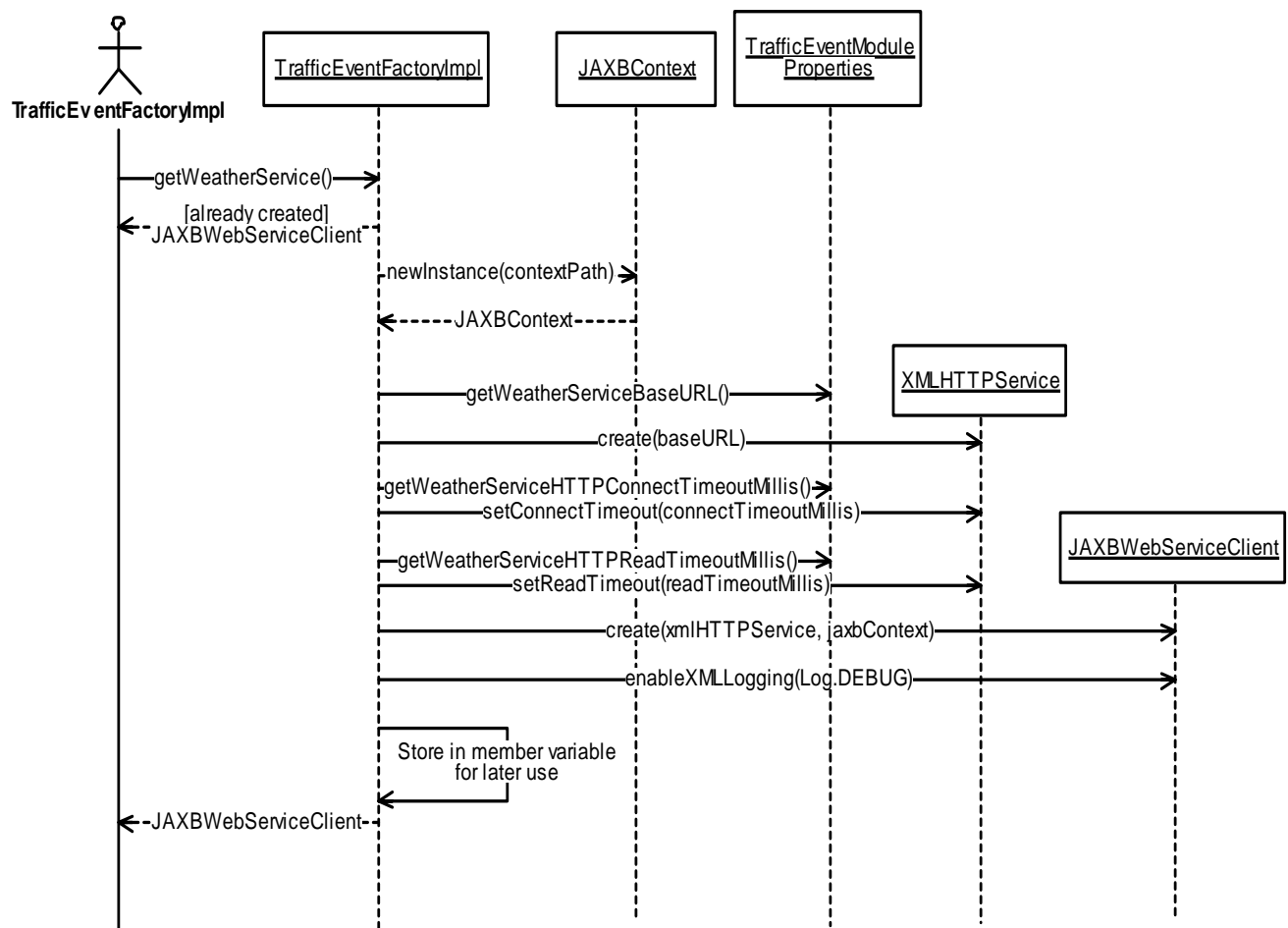
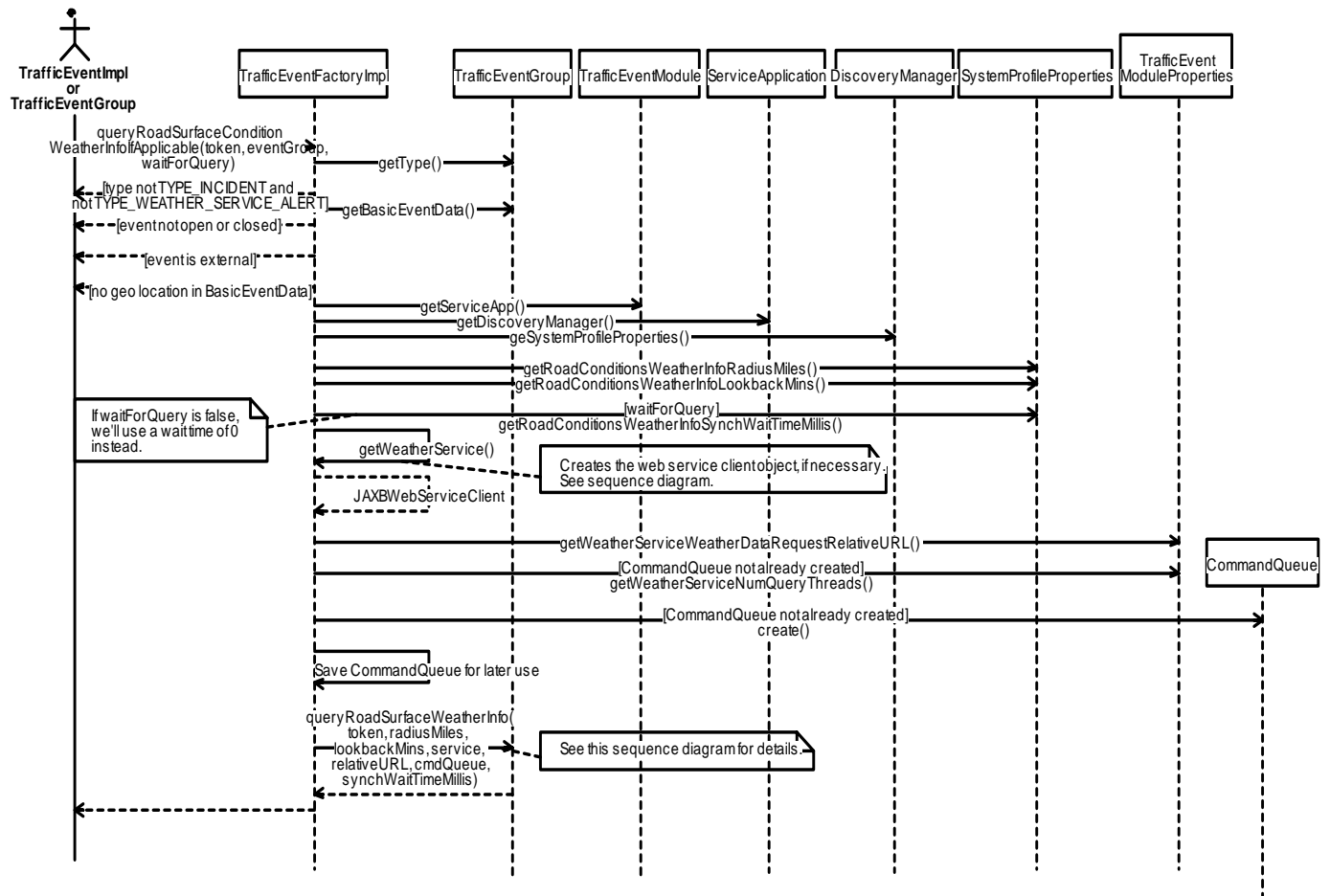


Figure 9-6. TrafficEventFactoryImpl:getWeatherService (Sequence Diagram)

### 9.3.2.3 TrafficEventFactoryImpl:queryRoadSurfaceConditionWeatherInfoIfApplicable (Sequence Diagram)

This diagram shows how the traffic event factory performs a road conditions weather info query. It returns and does nothing if the traffic event is not an incident or weather service alert, or if the event is pending or external, or if the traffic event does not have a geographic location. Otherwise, it gets the search radius, lookback time, and synch wait time from the System Profile. It gets (or creates) the JAXBWebServiceClient object (as shown on the getWeatherService sequence diagram). It creates a CommandQueue for handling weather queries, if one was not previously created. It then calls the TrafficEventGroup to perform the query (as shown on the queryRoadSurfaceWeatherInfo sequence diagram).



**Figure 9-7.**  
**TrafficEventFactoryImpl:queryRoadSurfaceConditionWeatherInfoIfApplicable (Sequence Diagram)**



### 9.3.2.4 TrafficEventGroup:close (Sequence Diagram)

This diagram shows the processing when a traffic event is closed. (NOTE - most of this is unchanged for R7, except for the weather query). If the event is Pending, an InvalidState exception is returned. If it is not external a resource check is performed, and if it is controlled by another operations center and the user does not have override rights, a ResourceControlConflict exception is returned. Otherwise, the BasicEventData fields are updated to mark the event as closed, and the values are saved to the database. The response plan items are revoked, removed, deactivated. CORBA events are pushed to inform other applications of the RPIs being deactivated and the event being closed. Log entries are added to the event history, comm log, and operations log. Finally, a call to query the road surface condition is made (if applicable), passing a "wait" parameter of false to avoid waiting for the query results on the calling thread, so the closing of the even is not delayed. The query will be handled on a background thread, and when it completes it will cause handleRoadSurfaceConditionWeatherInfo to be called (see that sequence diagram for details).

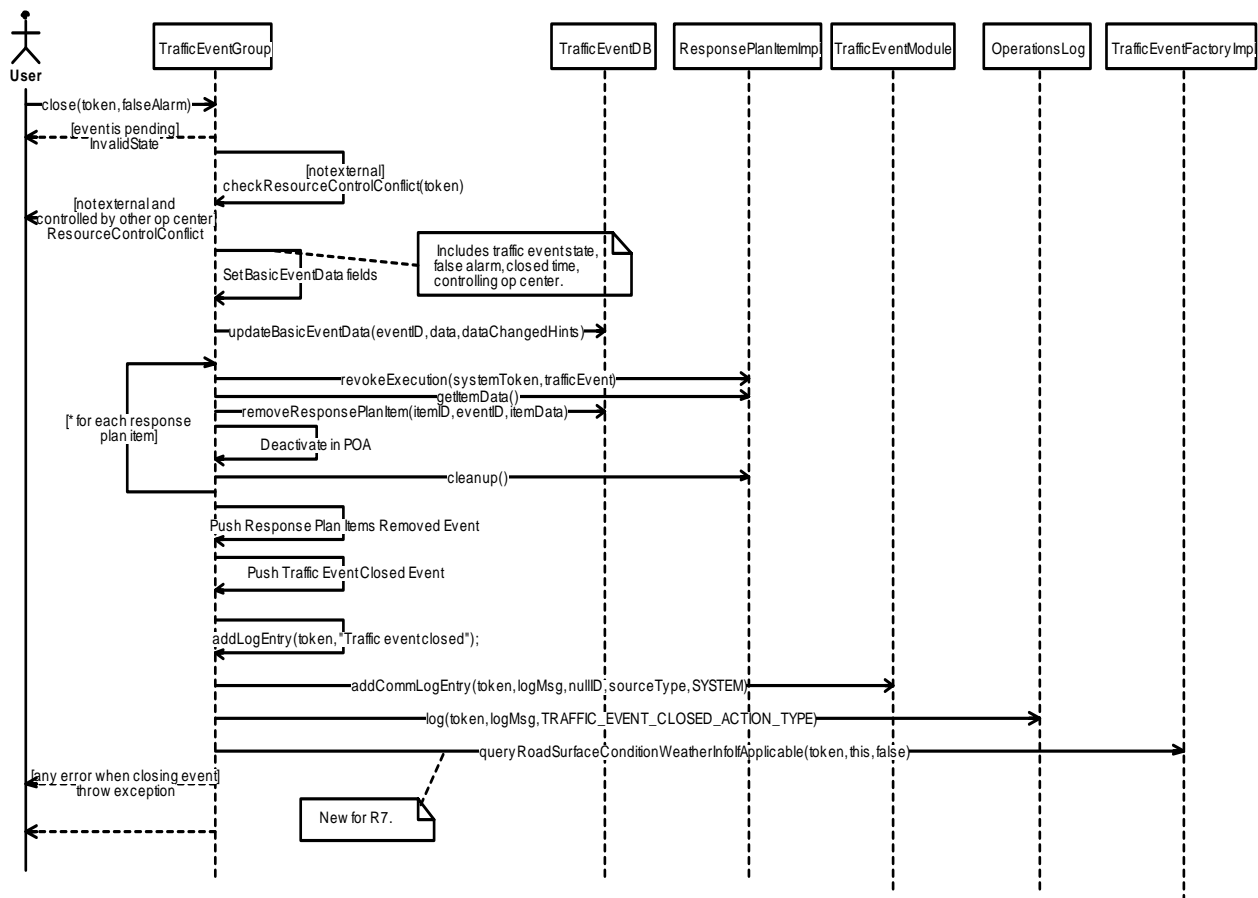
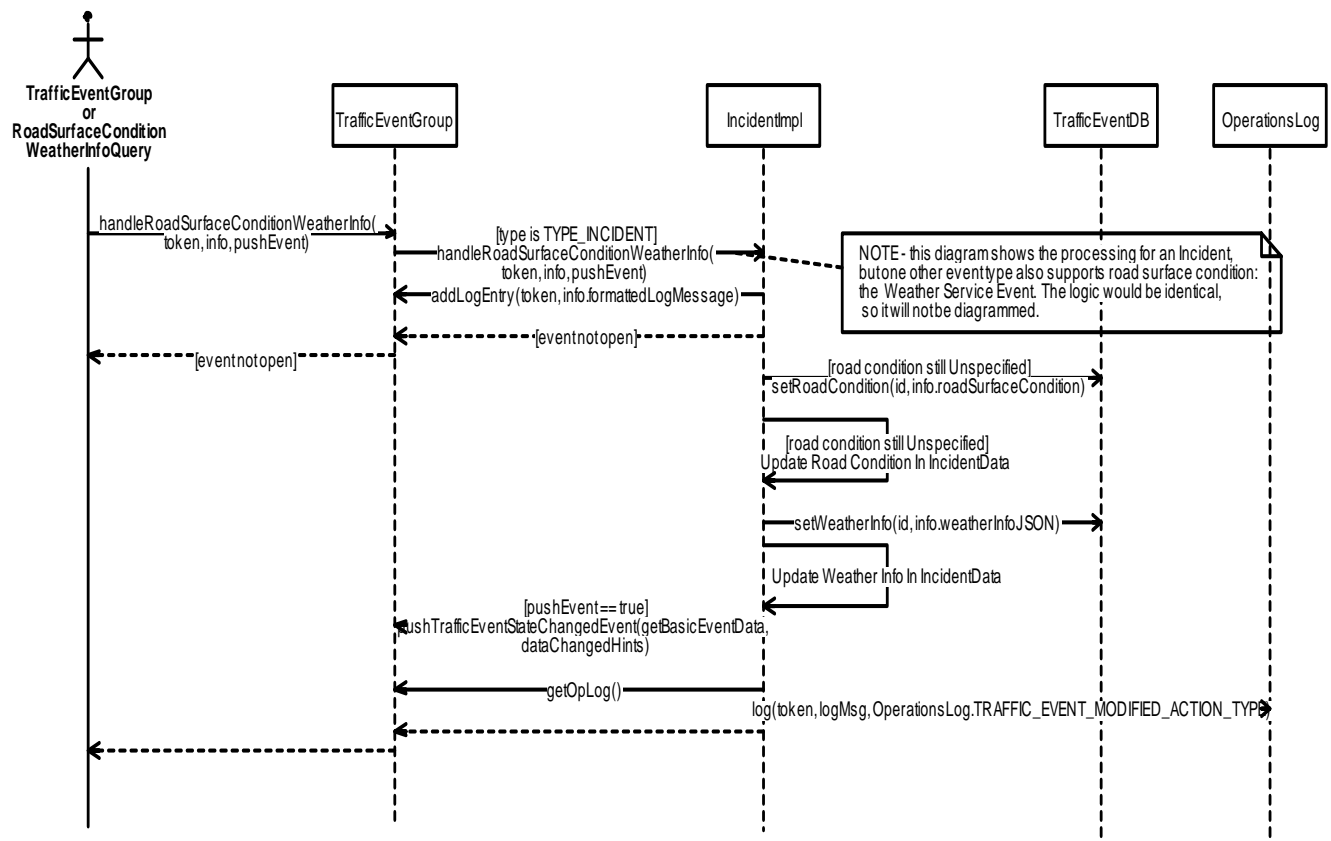


Figure 9-8. TrafficEventGroup:close (Sequence Diagram)

### 9.3.2.5 TrafficEventGroup:handleRoadSurfaceConditionWeatherInfo (Sequence Diagram)

This diagram shows how the results of the road surface condition weather info query are handled. The TrafficEventGroup will be called, either synchronously or asynchronously, to handle the results. If asynchronous, the pushEvent parameter will be true. The TrafficEventGroup will call the IncidentImpl to handle the data. It logs the weather data to the event history log. If the event is closed, it returns without further action. If the event is open, it calls the TrafficEventDB to set the road surface condition and weather info in the database, and then updates the fields in the BasicEventData object. If the pushEvent parameter was set to true, an event is pushed indicating the changed data. Finally an entry is added to the operations log before returning.

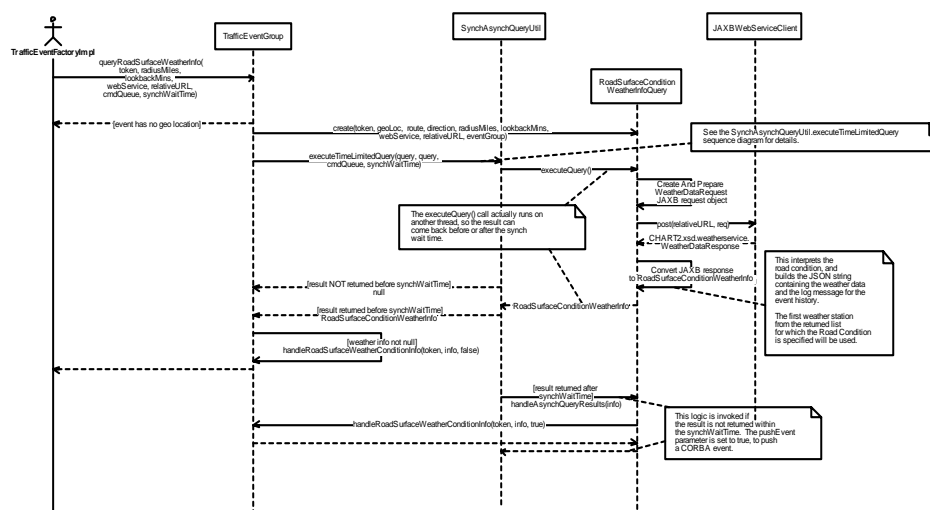


**Figure 9-9. TrafficEventGroup:handleRoadSurfaceConditionWeatherInfo (Sequence Diagram)**

### 9.3.2.6 TrafficEventGroup:queryRoadSurfaceWeatherInfo (Sequence Diagram)

This diagram shows how the query for road surface condition weather info will be made. The traffic event factory calls the traffic event group to initiate the query. The traffic event

group creates a RoadSurfaceConditionWeatherInfoQuery object which contains all of the data necessary to perform the query and to handle the results of the query. The query object is passed to the executeTimeLimitedQuery() method, which causes the query to be executed on another thread (see the SynchAsynchQueryUtil.executeTimeLimitedQuery sequence diagram for details). When the query is executed, a JAXB request object is created and posted to the Weather Service (web service), and a JAXB object is returned. The first weather station in the returned list for which a Road Condition is specified will be used. The data is then reduced to the information needed internally by the Traffic Event Service (i.e., a RoadSurfaceConditionWeatherInfo object). If the query completes before the specified synchronous wait time, the data is handled on the calling thread, so that the data will be included in traffic event creation. If the query takes longer than the specified wait time, the data will be handled asynchronously and will not show up immediately at event creation time, but will be pushed via a CORBA event. This mechanism allows a best effort to display the road conditions when the traffic event is first created, while ensuring that event creation will not be excessively slowed down if there is a problem querying the Weather Service.



**Figure 9-10. TrafficEventGroup:queryRoadSurfaceWeatherInfo (Sequence Diagram)**

## 9.4 Utility Package

### 9.4.1 Class Diagrams

#### 9.4.1.1 UtilityClasses3 (Class Diagram)

This Class Diagram shows various utility classes that are used by GUI and servers.

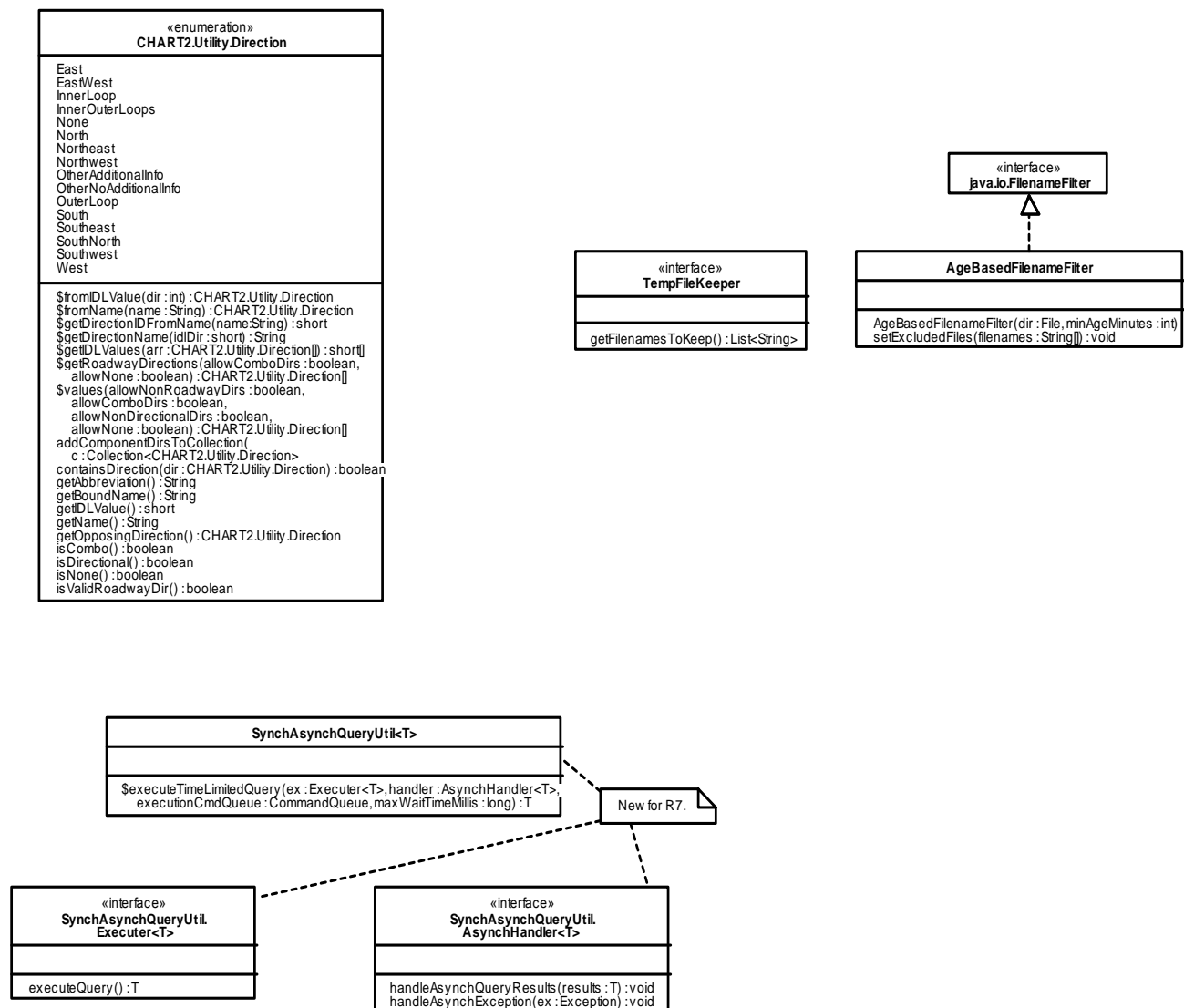


Figure 9-11. UtilityClasses3 (Class Diagram)

#### **9.4.1.1.1 AgeBasedFilenameFilter (Class)**

This class allows files that are older than the specified age to be accepted by the filter. This would typically be used to delete old files. Filenames to keep can be set into the filter, and these will not be accepted by the filter.

#### **9.4.1.1.2 CHART2.Utility.Direction (Class)**

This is an enumeration that mirrors the Common::Direction IDL enumeration, but provides additional functionality.

#### **9.4.1.1.3 java.io.FilenameFilter (Class)**

This interface is used to filter files by name.

#### **9.4.1.1.4 SynchAsynchQueryUtil. AsynchHandler<T> (Class)**

This interface allows an object to handle the data returned from a query asynchronously (i.e., not on the calling thread).

#### **9.4.1.1.5 SynchAsynchQueryUtil. Executer<T> (Class)**

This interface is implemented by an object that can execute a query. The type of data returned from the query is specified as a generic type. Any data needed for the query must be stored in the implementing object.

#### **9.4.1.1.6 SynchAsynchQueryUtil<T> (Class)**

This class performs time-limited synchronous queries, where it waits up to a specified maximum amount of time (on the calling thread) for the query to complete. If the query does not complete in that amount of time, the processing is handled asynchronously and the results are passed to a handler when the query does complete. This uses a parameterized type to allow generic query functionality.

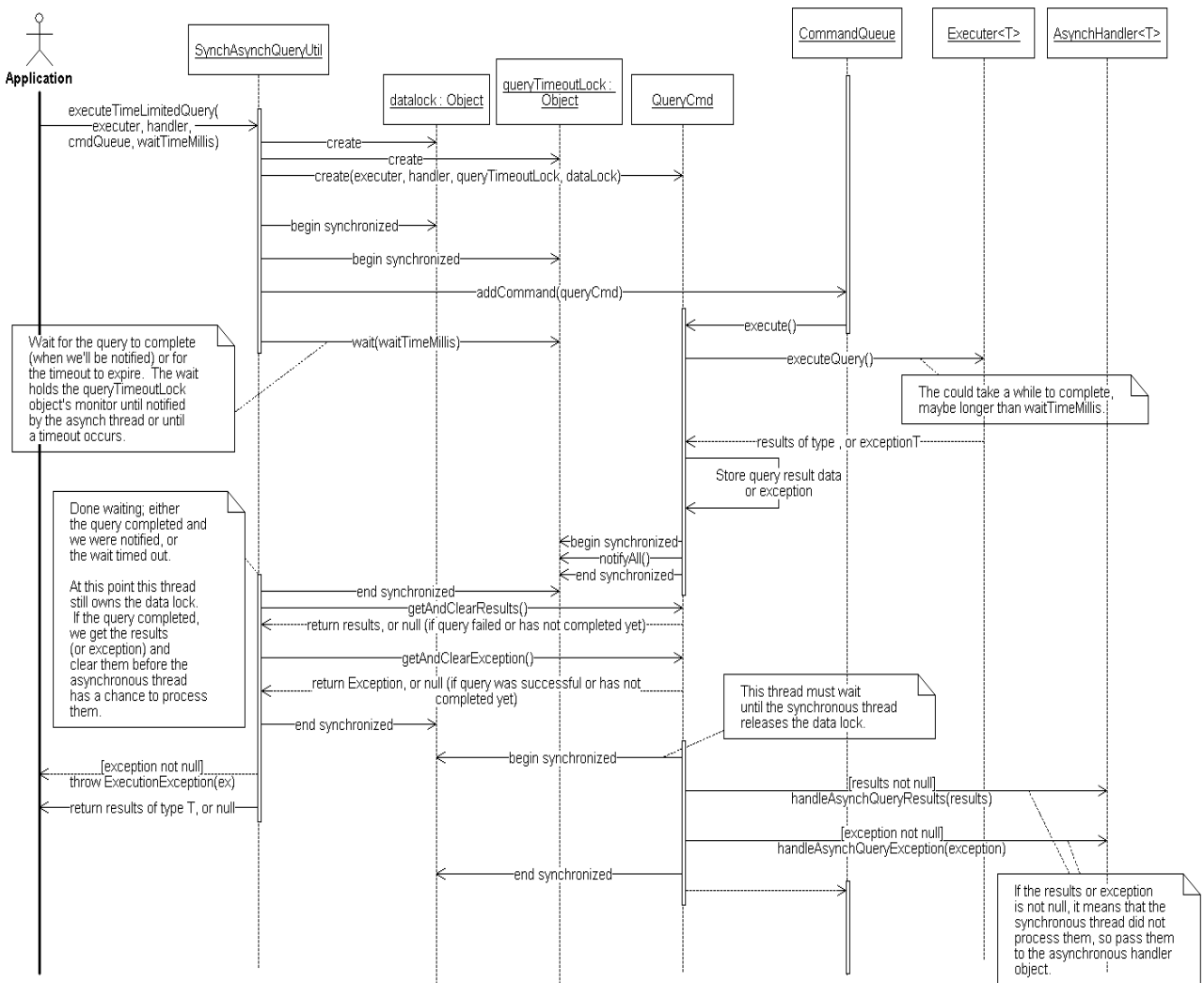
#### **9.4.1.1.7 TempFileKeeper (Class)**

This interface allows a class to "own" temporary files for the purpose of preventing them from being deleted by periodic cleanup functionality.

## 9.4.2 Sequence Diagrams

### 9.4.2.1 SynchAsynchQueryUtil:executeTimeLimitedQuery (Sequence Diagram)

This diagram shows the processing for a query that is allowed to take up to a limited amount of time to complete before being handled asynchronously. When `executeTimeLimitedQuery()` is called, an `Executer` and an `AsynchHandler` object are passed in, providing the app-level functionality for the query. A command queue is passed for executing the query, and a maximum wait time is passed. Two lock objects are created: one for the data, and one for the query timeout. The calling (synchronous) thread enters synchronized blocks for both of these locks, creates a command and adds it to the `CommandQueue`, and then waits for notification that the query has completed, or for the timeout to expire. The `CommandQueue` executes the query, and then stores the query result (or exception) in the command object for later use, notifies the synchronous thread to wake up, and then enters a synchronized block on the data lock. However, since the synchronous thread still has the data lock, the asynchronous thread is blocked until the synchronous thread extracts the query results (or exception) and clears them within the command object. The synchronous thread returns the results to the caller with the data (or null, if the query did not complete in time), or throws an exception to the caller. The asynchronous thread is then allowed to examine the results (or exception) stored in the command object. If they are not null at this point it is because the result (or exception) was not returned synchronously to the caller, so the `AsynchHandler` object is called to handle them.



**Figure 9-12. SynchAsynchQueryUtil:executeTimeLimitedQuery (Sequence Diagram)**

## 9.5 Webservices Weather Module Package

### 9.5.1 Class Diagrams

#### 9.5.1.1 WeatherModuleClasses (Class Diagram)

This diagram describes classes involved in the implementation of the CHART Weather Module.

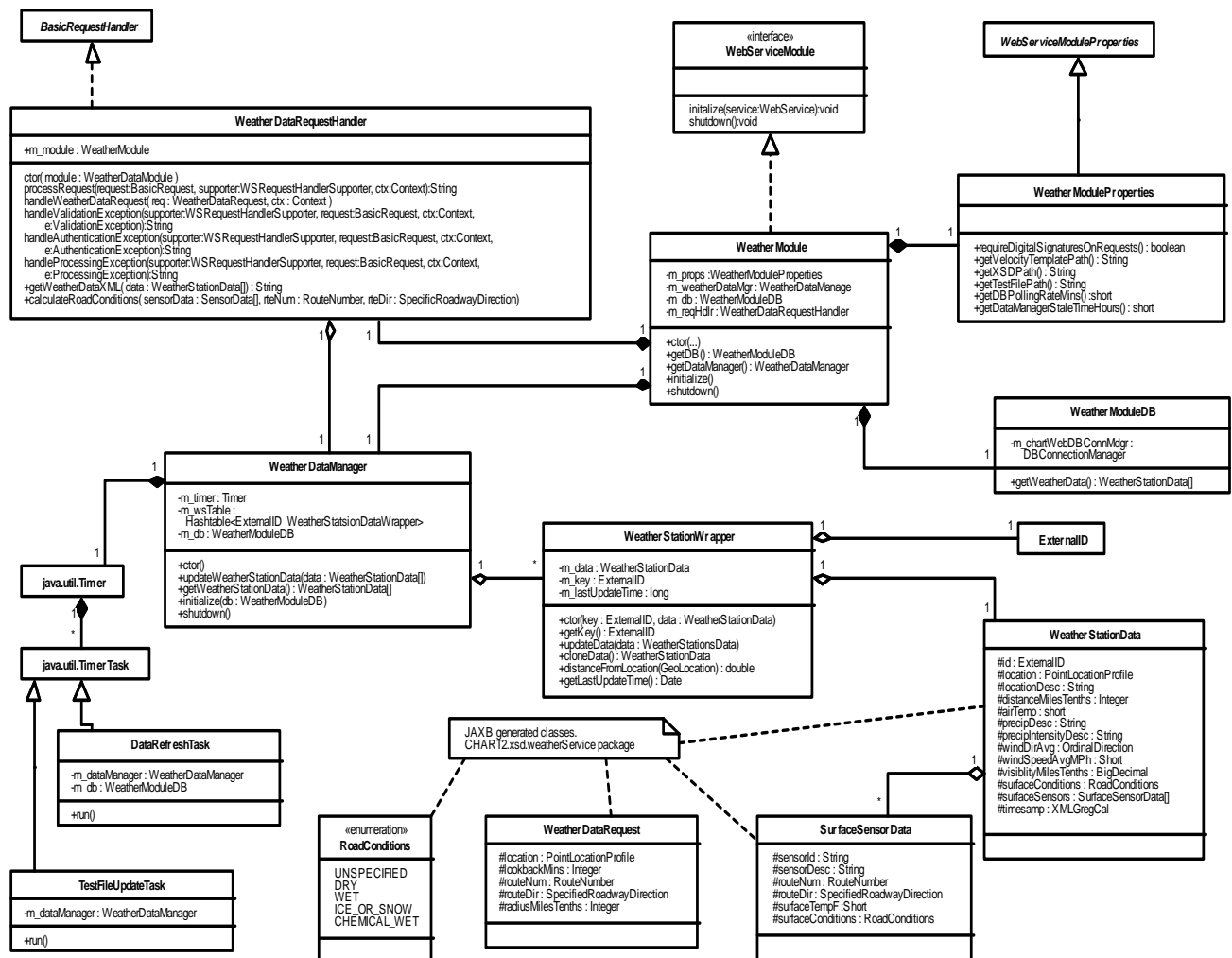


Figure 9-13. WeatherModuleClasses (Class Diagram)

##### 9.5.1.1.1 BasicRequestHandler (Class)

This abstract base class provides an implementation of the `WSRequestHandler.processRequest()` method that provides optional XML validation



against specified XSD files and optional digital signature verification as well. It is intended to be used by request handlers that plan to take XML in and return XML to the calling client.

#### **9.5.1.1.2 DataRefreshTask (Class)**

This TimerTask is used to periodically query the database containing weather data and using it to update the cache in the WeatherDataManager.

#### **9.5.1.1.3 ExternalID (Class)**

This object represents an External Object ID made up of 3 parts. A System ID, Agency ID, Object ID.

#### **9.5.1.1.4 java.util.Timer (Class)**

This class provides asynchronous execution of tasks that are scheduled for one-time or recurring execution.

#### **9.5.1.1.5 java.util.TimerTask (Class)**

This class is an abstract base class which can be scheduled with a timer to be executed one or more times.

#### **9.5.1.1.6 RoadConditions (Class)**

JAXB generated enum representing the road conditions values recognized in chart.

#### **9.5.1.1.7 SurfaceSensorData (Class)**

JAXB generated class (from the WeatherService.xsd) representing the data for a specific Surface Sensor..

#### **9.5.1.1.8 TestFileUpdateTask (Class)**

This Optional TimerTask is used to periodically read a test xml file used to provide input in a test mode. It can be used when it is not feasible or desirable to connect to a database.

#### **9.5.1.1.9 WeatherDataManager (Class)**

This class is responsible for maintaining the current cache of weather station data, and handling the processing involved when responding to request for data.

#### **9.5.1.1.10 WeatherDataRequest (Class)**

JAXB generated class (from the WeatherService.xsd) representing the request for weather data. It contains optional parameter such as location to consider, radius from location and lookback window.

#### **9.5.1.1.11 WeatherDataRequestHandler (Class)**

This class is the request handler that is responsible for handling Weather data requests. .

#### **9.5.1.1.12 WeatherModule (Class)**

This class is the pluggable web service module that provides Weather conditions lookup functionality.

#### **9.5.1.1.13 WeatherModuleDB (Class)**

This class manages connections to the database where weather data is retrieved from (currently the SCAN DB via the chart web db).

#### **9.5.1.1.14 WeatherModuleProperties (Class)**

This class extends the WebServiceModuleProperties and will provide all of the configurable property values for the WeatherModule.

#### **9.5.1.1.15 WeatherStationData (Class)**

JAXB generated class (from the WeatherService.xsd) representing the data for a specific WeatherStation.

#### **9.5.1.1.16 WeatherStationWrapper (Class)**

This class wraps a WeatherStationData object and provides accessor and convenience methods used during weather request data processing.

#### **9.5.1.1.17 WebServiceModule (Class)**

This interface defines the methods that each module must implement in order to run within the web service framework.

#### **9.5.1.1.18 WebServiceModuleProperties (Class)**

This abstract base class provides a base for WebServiceModule implementation classes to extend in order to get access to their configuration properties.

## 9.5.2 Sequence Diagrams

### 9.5.2.1 WeatherModule:initialize (Sequence Diagram)

This diagram describes the initialization of the WeatherModule class in the WeatherService. A module properties class is created followed by the creation of the WeatherModuleDB class. Currently this provides access the SCAN Data via the CHARTWeb DB. Next the WeatherDataManager is created and initialized. This class is responsible for maintaining a cache of Weather data making it available at any time for querying. Next the WeatherDataRequestHandler is created and registered with the WebService framework to allow the web service URL to start responding to requests.

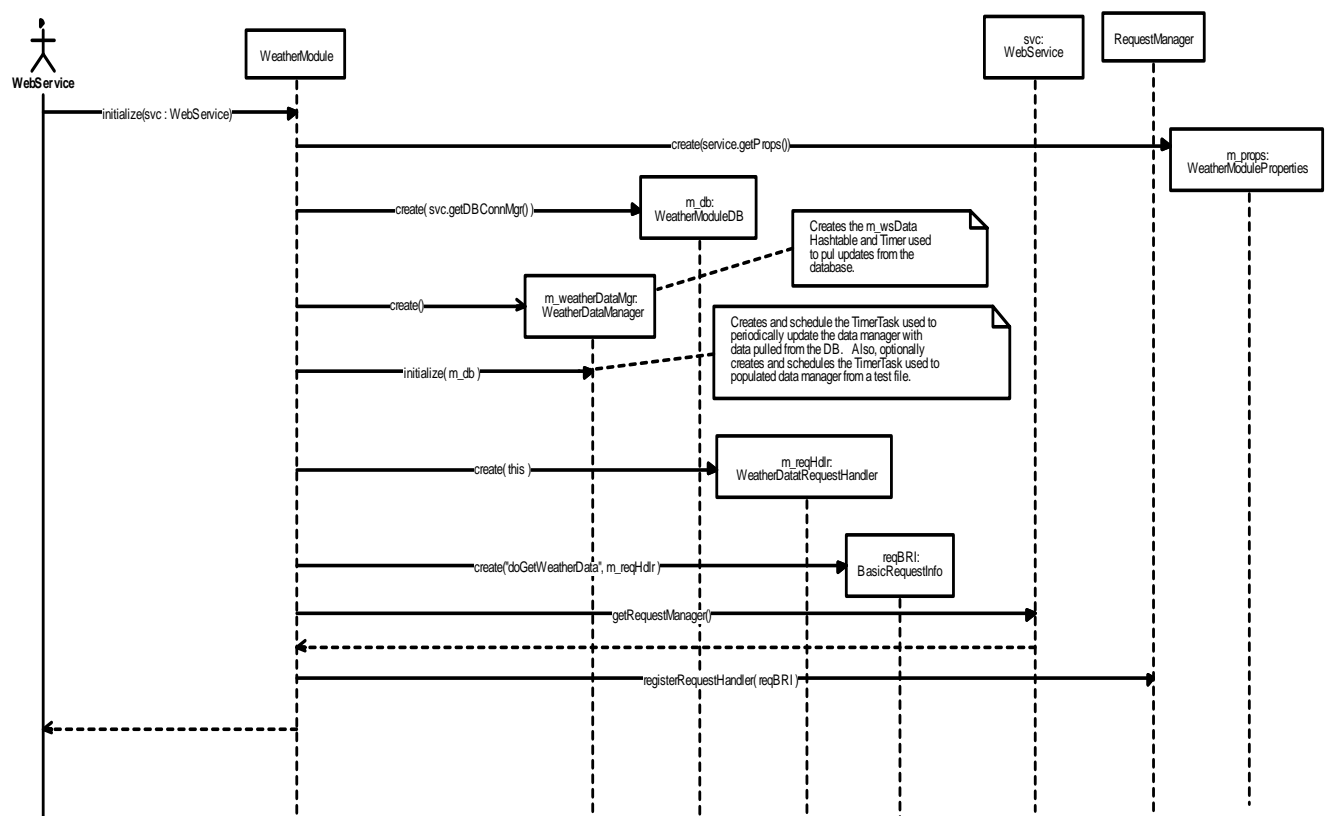
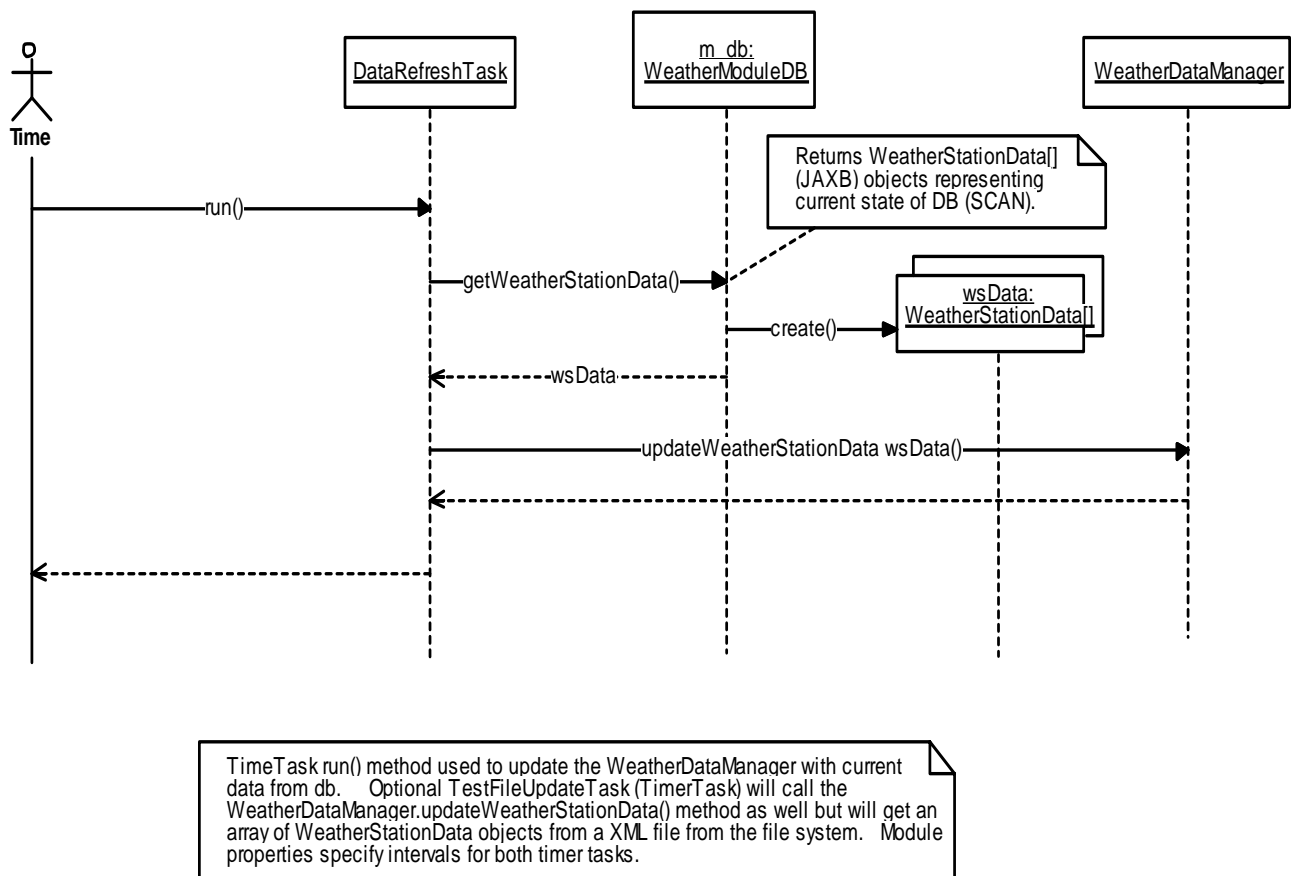


Figure 9-14. WeatherModule:initialize (Sequence Diagram)

### 9.5.2.2 DataRefreshTask:run (Sequence Diagram)

This diagram describes the DataRefreshTask.run() method (TimerTask) that will periodically query the CHARTWeb DB for SCAN weather data. The retrieved data is then passed to the WeatherDataManager.updateWeatherStationData() method to update its cache. The optional TestFileUpdateTask TimerTask will do similar processing except it will retrieve WeatherStationData periodically from an XML file in the file system.



**Figure 9-15. DataRefreshTask:run (Sequence Diagram)**

### 9.5.2.3 WeatherDataManager:updateWeatherStationData (Sequence Diagram)

This diagram describes the WeatherDataManager.updateWeatherStationData() method which is responsible for keeping the cache of weather data up to date. The method is passed an array of WeatherStationData objects. The method then loops thru the WeatherStationData objects and updates the Hashtable of WeatherStationDataWrapper objects by either updating an existing wrapper or creating a new one if needed. This method is currently called by two TimerTasks. The DataRefreshTask, which pulls current data from the SCAN DB, and the TestFileTask, which uses an externally supplied xml file to update the WeatherDataManager. The later is meant to be used for testing purposes when connecting to a db may not be desirable.

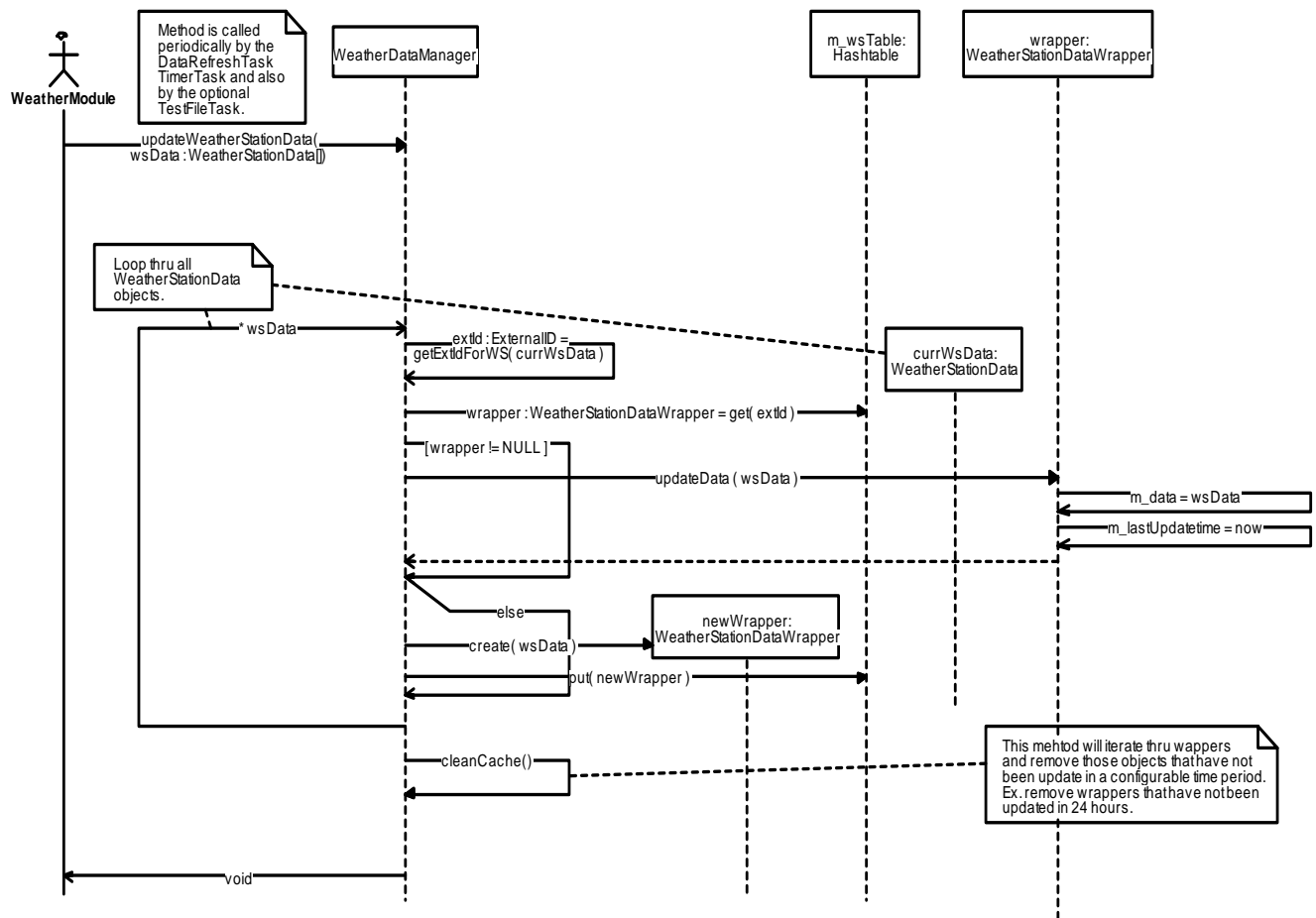
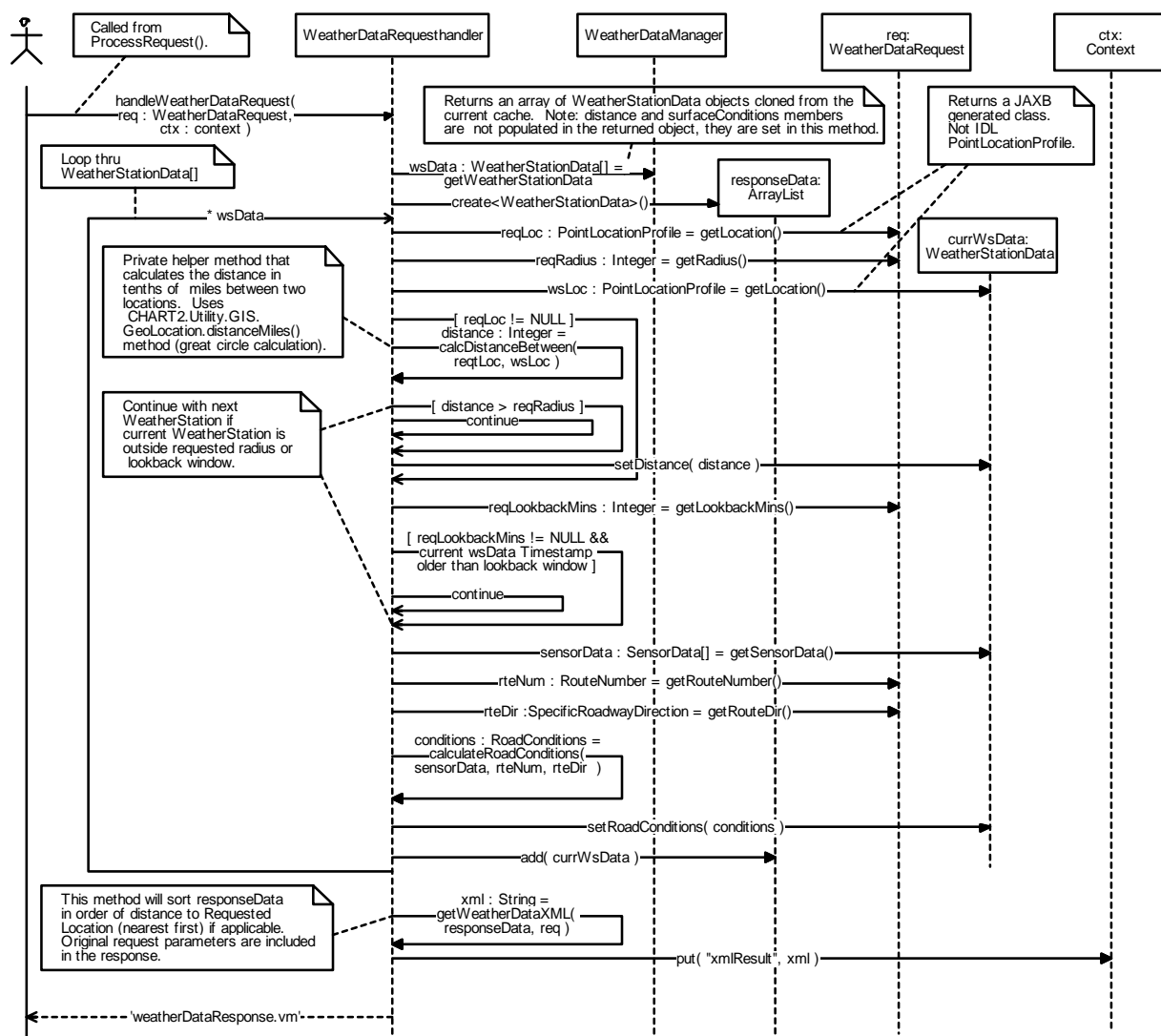


Figure 9-16. WeatherDataManager:updateWeatherStationData (Sequence Diagram)

#### 9.5.2.4 WeatherDataRequestHandler:handleWeatherDataRequest (Sequence Diagram)

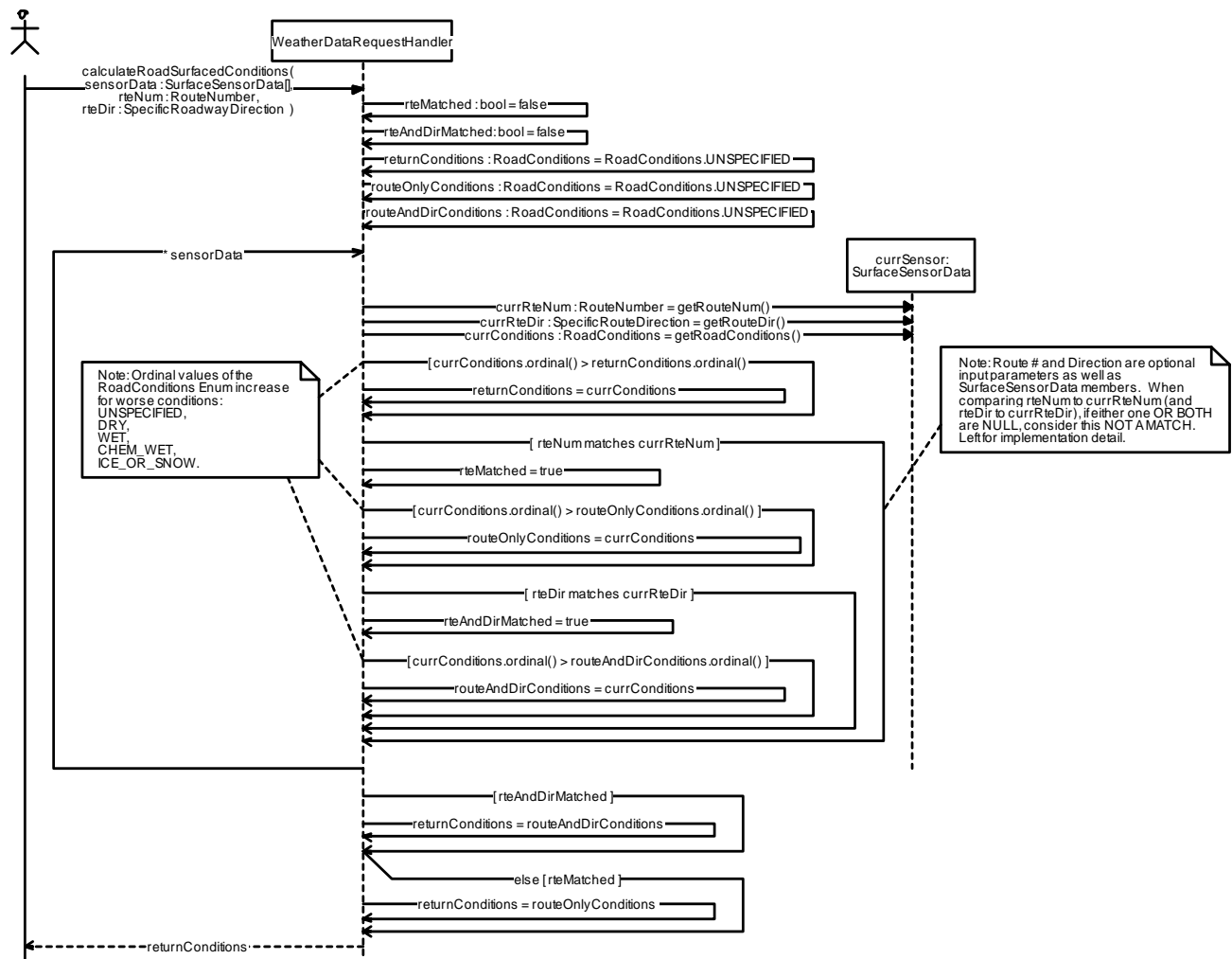
This diagram describes the WeatherDataRequestHandler.handleWeatherDataRequest() method which is call by the WeatherDataRequestHandler.processRequest() method when the Webservice receives a request for weather data. The method retrieves a list of cloned WeatherStationData from the WeatherDataManager. As the method loops thru the WeatherStationData objects it will ignore objects based on location and look back if specified in request. If route number and direction is specified in request, an attempt is made to determine a surface conditions value for the WeatherStation. When all WeatherStationData objects have been processed, XML is generated based on the remaining WeatherStationData objects. The XML string is added to the Context passed in and the path for the velocity template used to respond to the request is returned.



**Figure 9-17. WeatherDataRequestHandler:handleWeatherDataRequest (Sequence Diagram)**

### 9.5.2.5 WeatherDataRequestHandler:calculateRoadSurfaceConditions (Sequence Diagram)

This diagram describes the WeatherDataRequestHandler.calculateSurfaceConditions() method which is used to determine a WeatherStation level Surface Conditions value based on a set of SurfaceSensorData objects and optional route number and direction arguments passed in. First preference is to select sensors that match route number and direction. Second preference is to select sensors that match just route number. Third preference is to select all sensors at the weather station. Returns the most adverse surface condition for all selected surface sensors according to this list of increasing adversity: UNSPECIFIED, DRY, WET, CHEMICALLY WET, ICE OR SNOW, Defaults to the UNSPECIFIED surface condition.



**Figure 9-18. WeatherDataRequestHandler:calculateRoadSurfaceConditions (Sequence Diagram)**



## **10 Deprecated Functionalities**

---

The following functions have been deprecated due to CHART R7 / Mapping R6 changes.

### **10.1 CHART Device Editor**

#### **10.1.1 View, Add, Update, Remove CHART Devices**

In Release 7, the Integrated Map in the CHART application will start handling the mapping of CHART Devices (DMS, HAR, SHAZAM, CAMERA, DETECTOR); Device viewing, adding, updating, and removing will no longer be available in the Mapping CHART Device Editor.

## 11 Mapping To Requirements

The following table shows how the requirements in the CHART R7 Requirements document map to design elements contained in this design.

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR 1	ADMINISTER SYSTEMS AND EQUIPMENT		N/A	N/A
SR1.4	MANAGE CHART CONTROL		N/A	N/A
SR1.4.2	PERFORM SHIFT HAND-OFF (INCOMING) AND VIEW OPERATIONS CENTER HOME PAGE		N/A	N/A
SR1.4.2.2	The system shall allow a system administrator to enter a message of the day to be displayed within the shift hand-off report.	Shift Handoff	N/A	N/A
SR1.4.3	MAINTAIN SHIFT HAND OFF REPORT		N/A	N/A
SR1.4.3.1	The system shall allow the user to enter notes and review other users' notes related to shift or Center activities.	Shift Handoff	N/A	N/A
SR1.4.3.1.1	The system shall provide an online Shift Hand Off Report supporting the freeform entry of text.	Shift Handoff	N/A	N/A
SR1.4.3.1.2	A sufficiently privileged user shall be able to enter information into the Shift Hand Off Report.	Shift Handoff	N/A	N/A

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.4.3.1.3	The Shift Hand Off Report shall be viewable by any user.	Shift Handoff	N/A	N/A
SR1.4.3.2	The system shall automatically assign and display the userid and date/timestamp for each Shift Hand Off Report entry.	Shift Handoff	N/A	N/A
SR1.4.3.3	The system shall allow the user to format content for text appearance and tables without having to enter HTML commands.	Shift Handoff	N/A	N/A
SR1.4.3.5	The system shall automatically flag and/or remove out-dated information.	Shift Handoff	N/A	N/A
SR1.4.3.7	The system shall allow the organization of the data within the shift hand-off report to be customized.	Shift Handoff	N/A	N/A
SR1.4.3.8	The system shall support a user hierarchy to control the allowable actions for each user.	Shift Handoff	N/A	N/A
SR1.4.3.8.1	A user level shall exist that allows users to add, edit, and remove their own entries, but not other user's entries.	Shift Handoff	N/A	N/A
SR1.4.3.8.2	A user level shall exist that allows users to add, edit, and remove their own entries and entries made by other users.	Shift Handoff	N/A	N/A
SR1.4.3.8.3	A user level shall exist that allows the user to perform all features available in the shift hand off report (administrator).	Shift Handoff	N/A	N/A

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.4.3.9	The system shall include a calendar within the shift hand off report.	Shift Handoff	N/A	N/A
SR1.4.3.9.1	The calendar in the shift hand off report shall be capable of displaying events.	Shift Handoff	N/A	N/A
SR1.4.3.9.2	The calendar in the shift hand off report shall allow suitably privileged users to add, edit, and remove entries from the calendar.	Shift Handoff	N/A	
SR1.4.3.9.3	The system shall allow any user to view the calendar in the shift hand off report.	Shift Handoff	N/A	N/A
SR1.4.3.10	The system shall allow static reference material to be included in the shift handoff report.	Shift Handoff	N/A	N/A
SR1.5	INSTALL AND MAINTAIN DEVICES		N/A	N/A
SR1.5.2	PUT EQUIPMENT/ DEVICES ON-LINE		N/A	N/A
SR1.5.2.1	The system shall allow the user with appropriate rights to select (or modify) the equipment device parameters.		N/A	N/A
SR1.5.2.1.17	The system shall support configuration parameters for TSS (Traffic Sensor System) devices (detectors).		N/A	N/A

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.5.2.1.17.13	Specify RTMS Configuration		N/A	N/A
SR1.5.2.1.17.13.6	The system shall allow the user to specify the grouping of the RTMS detection zones into one or more logical zone groups.		N/A	N/A
SR1.5.2.1.17.13.15	The system shall allow a user with the "configure TSS" functional right to edit the map display properties for any RTMS that has a defined location (lat/lon).	Map	R7HighLevel.ConfigureTSS ConfigureTSS.EditMapDisplayOptions	chartlite.servlet.tss_classes CD chartlite.servlet.tss:getEditTSSMapDisplayOptionsForm SD chartlite.servlet.tss:processUpdateMapDisplayOptions SD
SR1.5.2.1.17.13.15.1	The system shall allow a user with the configure TSS functional right to set the map display options for the RTMS in any mode (online, offline or maintenance mode).	Map	R7HighLevel.ConfigureTSS ConfigureTSS.EditMapDisplayOptions	chartlite.servlet.tss:getEditTSSMapDisplayOptionsForm SD chartlite.servlet.tss:processUpdateMapDisplayOptions SD
SR1.5.2.1.17.13.15.2	The system shall allow the user to specify a primary display bearing for the TSS as a value between 0 and 359 degrees, with a bearing of 0 degrees indicating due east, with the bearing growing counter-clockwise. (So a bearing of 90 indicates due north, 180 indicates due west and 270 indicates due south.)	Map	R7HighLevel.ConfigureTSS ConfigureTSS.EditMapDisplayOptions ConfigureTSS.SetBearing	TSSManagement CD GUITSSDataClasses CD chartlite.servlet.tss:processUpdateZoneGroupDisplayDirection SD Screenshot: HMI Figure 4-3
SR1.5.2.1.17.13.15.2.1	When initially created a TSS will not have a defined bearing.	Map	R7HighLevel.ConfigureTSS ConfigureTSS.EditMapDisplayOptions ConfigureTSS.SetBearing	TSSManagement CD GUITSSDataClasses CD

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.5.2.1.17.13 .15.3	A zone group shall include a flag that indicates how the zone group should be displayed on maps.	Map	R7HighLevel.Configure TSS ConfigureTSS.EditMap DisplayOptions ConfigureTSS.SetZone GroupDisplayDirection	TSSManagement CD GUITSSDataClasses CD Screenshot: HMI Figure 4-4
SR1.5.2.1.17.13 .15.3.1	The system shall allow a user to specify that a zone group should be displayed on maps using an arrow that points in the direction of the TSS bearing.	Map	R7HighLevel.Configure TSS ConfigureTSS.EditMap DisplayOptions ConfigureTSS.SetZone GroupDisplayDirection	TSSManagement CD GUITSSDataClasses CD Screenshot: HMI Figure 4-4
SR1.5.2.1.17.13 .15.3.2	The system shall allow a user to specify that a zone group should be displayed on maps using an arrow that points in the direction opposite (180 degrees opposed) to the TSS bearing.	Map	R7HighLevel.Configure TSS ConfigureTSS.EditMap DisplayOptions ConfigureTSS.SetZone GroupDisplayDirection	TSSManagement CD GUITSSDataClasses CD Screenshot: HMI Figure 4-4
SR1.5.2.1.17.13 .15.3.3	The system shall allow the user to indicate that the zone group should not be displayed on maps.	Map	R7HighLevel.Configure TSS ConfigureTSS.EditMap DisplayOptions ConfigureTSS.SetZone GroupDisplayDirection	TSSManagement CD GUITSSDataClasses CD Screenshot: HMI Figure 4-4
SR1.5.2.1.17.13 .15.3.4	The system shall warn a user who is changing a zone group from not displayable on maps to displayable on maps that the zone group name may be displayed on the Internet map. This warning shall include the currently configured name of the zone group.	Map	R7HighLevel.Configure TSS ConfigureTSS.EditMap DisplayOptions ConfigureTSS.SetZone GroupDisplayDirection	Screenshot: HMI Figure 4-5
SR1.5.2.1.17.13 .15.4	The system shall allow a user with the configure TSS functional right to set the display order for each zone group relative to other zone groups of the TSS with the same display bearing.	Map	R7HighLevel.Configure TSS ConfigureTSS.EditMap DisplayOptions ConfigureTSS.SetZone GroupDisplayOrder	TSSManagement CD GUITSSDataClasses CD Screenshot: HMI Figure 4-6

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.5.2.1.17.13.16	The system shall allow an administrator to configure the number of pixels offset that should be used for displaying TSS arrows at each map zoom scale.	Map	R7HighLevel.ConfigureTSS ConfigureTSS.SetOffset	
SR1.5.2.1.18	The system shall support setting configuration parameters for Cameras.		N/A	N/A
SR1.5.2.1.18.7	Edit Camera Configuration		R7CameraUses.ucd	
SR1.5.2.1.18.7.4	The system shall allow a suitably privileged user to edit the settings for an existing NTCIP camera, as listed in the Specify Video Source Attributes, Specify Basic Camera Attributes, and Specify Controllable Camera Attributes requirements, except for the No Video Available flag.	NTCIP Camera	R7CameraUses.ucd	CameraControlModule:SetCameraConfiguration.etd
SR1.5.2.1.18.12	Add / Copy NTCIP Camera	NTCIP Camera	R7CameraUses.ucd	CameraControlModule:AddCamera.etd
SR1.5.2.1.18.12.1	The system shall allow a suitably privileged user to add an NTCIP camera to the system.	NTCIP Camera	R7CameraUses.ucd	CameraControlModule:AddCamera.etd
SR1.5.2.1.18.12.2	The system shall allow the user to specify the attributes listed under the Edit NTCIP Camera Configuration requirements when adding an NTCIP camera.	NTCIP Camera	R7CameraUses.ucd	CameraControlModule:AddCamera.etd
SR1.5.2.1.18.12.3	The system shall require the user to choose a factory site when adding an NTCIP camera.	NTCIP Camera	R7CameraUses.ucd	CameraControlModule:AddCamera.etd
SR1.5.2.1.18.12.4	The system shall allow the user to pre-populate the configuration settings for creating a new NTCIP camera using the settings of an existing NTCIP camera.	NTCIP Camera	R7CameraUses.ucd	CameraControlModule:AddCamera.etd

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.5.2.10	The system shall allow the user with appropriate rights to put equipment on-line in CHART.		N/A	N/A
SR1.5.2.10.12	The system shall allow a suitably privileged user to put a camera online.		N/A	N/A
SR1.5.2.10.12.2	The system shall allow a suitably privileged user to put a COHU camera online.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR1.5.2.10.12.3	The system shall allow a suitably privileged user to put an NTCIP camera online.	NTCIP Camera	ManageCamera.ucd	
SR1.5.2.12	The system shall allow the suitably privileged to take a device offline		N/A	N/A
SR1.5.2.12.7	The system shall allow a suitably privileged user to take a camera offline.		N/A	N/A
SR1.5.2.12.7.2	The system shall allow a suitably privileged user to take a COHU camera offline.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR1.5.2.12.7.3	The system shall allow a suitably privileged user to take an NTCIP camera offline.	NTCIP Camera	ManageCamera.ucd	CameraControlModule:TakeCameraOffline.etd
SR1.5.3	PERFORM ROUTINE MAINTENANCE. The system shall allow the user with appropriate rights to view the device status, and know why it's not on-line (including the key trouble ticket information) and know the problem is being addressed. The system shall also allow the user to take the device offline of maintenance or other adjustments including resetting the controller. Suggestion/example to be validated: e.g., integrate device		N/A	N/A



Tag	Requirement	Feature	Use Cases	Other Design Elements
	maintenance web pages with CHART.			
SR1.5.3.11	The system shall allow a suitably privileged user to poll a camera for its current status if it is not offline.		N/A	N/A
SR1.5.3.11.2	The system shall allow a suitably privileged user to poll a COHU camera for its current status.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR1.5.3.11.3	The system shall allow a suitably privileged user to poll an NTCIP camera for its current status.	NTCIP Camera	ManageCamera.ucd ManageCameraControl.ucd	CameraControlModule.cad NTCIPCameraProtocolHdlr:poll.etd
SR1.5.4	RESPOND TO EQUIPMENT/ DEVICE OUTAGE.The system shall allow the user with appropriate rights to notify maintenance personnel of an equipment outage that they have detected (or has been brought to their attention).		N/A	N/A
SR1.5.4.7	The system shall generate a Device Failure Alert for all DMSs and TSSs capable of reporting that they are experiencing a hardware failure	NTCIP Camera	(Not a new requirement, just reworded. Used to say "all devices" which was not and is not accurate.)	
SR1.5.9	View Devices On Map		N/A	N/A
SR1.5.9.5	The system shall allow a suitably privileged user to view TSSs on the map.	Map	R7HighLevel.ViewTSS OnMap MapAndGISUses.View DevicesOnMap	MapClasses CD MapReqHdlr:getHomePageMapDataJSON SD Screenshot: HMI Figure 4-9 Screenshot: HMI Figure 4-10
SR1.5.9.5.1	The system shall display TSS devices on a set of map layers that contain only TSSs.	Map	R7HighLevel.ViewTSS OnMap MapAndGISUses.View DevicesOnMap	MapReqHdlr:getHomePageMapDataJSON SD MapReqHdlr:addJSONFeaturesForTSSLayers SD Screenshot: HMI Figure 4-8

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.5.9.5.1.1	The TSS map layer shall have a child layer that shows only CHART system TSSs.	Map	R7HighLevel.ViewTSS OnMap	MapReqHdlr:getHomePageMapDataJSON SD MapReqHdlr:addJSONFeaturesForTSSLayers SD Screenshot: HMI Figure 4-8
SR1.5.9.5.1.2	The TSS map layer shall have a child layer for each CHART organization that owns at least one external TSS	Map	R7HighLevel.ViewTSS OnMap	MapReqHdlr:getHomePageMapDataJSON SD MapReqHdlr:addJSONFeaturesForTSSLayers SD Screenshot: HMI Figure 4-8
SR1.5.9.5.2	The system shall allow the user to click on a TSS in the map to display a subset of the available information for the TSS.	Map	MapDeviceAndTraffic EventUses.UseTSSFro mMap	Screenshot: HMI Figure 4-10 MapReqHdlr:addJSONFeaturesForTSSLayers SD
SR1.5.9.5.2.4	The map callout shall include configuration information and current traffic parameters for each zone group.	Map	MapDeviceAndTraffic EventUses.UseTSSFro mMap	Screenshot: HMI Figure 4-10 MapReqHdlr:addJSONFeaturesForTSSLayers SD
SR1.5.9.5.2.4.1	The map callout shall include the name of each configured zone group.	Map	MapDeviceAndTraffic EventUses.UseTSSFro mMap	Screenshot: HMI Figure 4-10
SR1.5.9.5.2.4.2	The map callout shall include the current zone group volume, speed and occupancy if the currently logged in user has the view detailed VSO functional right.	Map	MapDeviceAndTraffic EventUses.UseTSSFro mMap	Screenshot: HMI Figure 4-10 MapReqHdlr:addJSONFeaturesForTSSLayers SD
SR1.5.9.5.2.4.3	The map callout shall include the current zone group speed summary if the currently logged in user has the view summary VSO functional right and does not have the view detailed VSO functional right.	Map	MapDeviceAndTraffic EventUses.UseTSSFro mMap	MapReqHdlr:addJSONFeaturesForTSSLayers SD

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.5.9.5.4	The system shall use a graphic to indicate the current status of a TSS on the map provided that the TSS has a defined lat/lon location.		MapDeviceAndTraffic EventUses.UseTSSFromMap	Screenshot: HMI Figure 4-9
SR1.5.9.5.4.1	If a the user has rights to see at least summary VSO data for the TSS and the TSS has a defined location (lat/lon), a defined bearing, at least one defined zone group that is displayable on maps, and is online (and not comm. failed, comm. marginal, or hardware failed) the system will show a graphic that uses the bearing and a colored arrow per zone group to depict current speed information for the TSS.	Map	MapDeviceAndTraffic EventUses.UseTSSFromMap	Screenshot: HMI Figure 4-9
SR1.5.9.5.4.1.1	The arrow for each zone group shall be colored to indicate the current speed range for that zone group.	Map	MapDeviceAndTraffic EventUses.UseTSSFromMap	Screenshot: HMI Figure 4-9
SR1.5.9.5.4.1.2	The system shall position the zone group arrows based on the configured zone group display order per direction, with lower numbers closer to center. (Starting at the location of the TSS, zone groups with a lower display order will appear first and zone groups with higher display orders will appear further away from the TSS lat/lon position.)	Map	MapDeviceAndTraffic EventUses.UseTSSFromMap	Screenshot: HMI Figure 4-7
SR1.5.9.5.4.2	If a TSS does not have a defined bearing, has no defined zone groups that are displayable on maps, is not online, or is online and comm. failed, comm. marginal, or hardware failed, the system will show a graphic on the map that matches the graphic in the device list for that particular TSS.	Map	MapDeviceAndTraffic EventUses.UseTSSFromMap	Screenshot: HMI Figure 4-9

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.5.10	Verify Device Compatibility		N/A	
SR1.5.10.2	The system shall provide a stand alone tool to allow Camera suppliers to test if an NTCIP Camera is compatible with the CHART system.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.1	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Pan feature operates properly on an NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.1.1	The NTCIP Camera Compatibility Tester shall allow the user to test that the CHART Camera Pan feature can pan the camera at a specific user-configured speed.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.1.2	The NTCIP Camera Compatibility Tester shall allow the user to test the ability for CHART to query zoom level and use it to automatically set pan speed on an NTCIP Camera to a reasonable value. ("Reasonable" means not intolerably slow nor uncontrollably fast.)	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.1.2.1	The computed NTCIP Camera pan speed value shall be interpolated along a linear scale within user-configured minimum and maximum pan speeds, as discussed in SR3.6.1.4.3.6.1.1.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.2	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Tilt feature operates properly on an NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.5.10.2.2.1	The NTCIP Camera Compatibility Tester shall allow the user to test that the CHART Camera Tilt feature can tilt the camera at a specific user-configured speed.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.2.2	The NTCIP Camera Compatibility Tester shall allow the user to test the ability for CHART to query zoom level and use it to automatically set tilt speed on an NTCIP Camera to a reasonable value. ("Reasonable" means not intolerably slow nor uncontrollably fast.)	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.2.2.1	The computed NTCIP Camera tilt speed value shall be interpolated along a linear scale within user-configured minimum and maximum tilt speeds, as discussed in SR3.6.1.4.3.6.1.2.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.3	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Zoom feature operates properly on an NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.4	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Auto Focus feature operates properly on an NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.5	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Adjust Focus feature operates properly on an NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.6	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Auto Iris feature operates properly on an NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.5.10.2.7	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Adjust Iris feature operates properly on an NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.8	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Set Preset feature operates properly on a NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.9	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Go to Preset feature operates properly on an NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.10	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Set Default Title Line one feature operates properly on an NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.11	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Default Title line two operates properly on an NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.12	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Camera Set Power On/Off feature operates properly on an NTCIP Camera.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.13	The NTCIP Camera Compatibility Tester shall provide output that shows the user the results of the tests that are run.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.13.1	The NTCIP Camera Compatibility Tester shall allow the user to save the test results to a file.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.5.10.2.14	The NTCIP Camera Compatibility Tester shall support connecting to the camera being tested via a direct RS232 connection.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.15	The NTCIP Camera Compatibility Tester shall support connecting to the camera being tested via a TCP/IP (network) connection.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.16	The NTCIP Camera Compatibility Tester shall allow the user to configure the communication settings used by the tester.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.16.1	The NTCIP Camera Compatibility Tester shall allow configuration of the following RS232 communication settings: Comm Port Name, Baud Rate, Data Bits, Parity, Stop Bits, and Flow Control.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.16.2	The NTCIP Camera Compatibility Tester shall allow configuration of the following TCP/IP communication settings: IP Address and Port.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.16.3	The NTCIP Camera Compatibility Tester shall allow configuration of the following general communication settings: Drop Address (Camera Number), SNMP Community String, HDLC Framing Required Flag, Initial Receive Timeout, Inter-character Receive Timeout, Total Receive Duration.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.17	The NTCIP Camera Compatibility Tester shall allow the user to configure the camera settings used by the tester.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR1.5.10.2.17.1	The NTCIP Camera Compatibility Tester shall allow configuration of the following camera settings: Minimum Pan Speed, Maximum Pan Speed, Minimum Tilt Speed, Maximum Tilt Speed, Zoom Speed, Focus Speed.	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR1.5.10.2.18	The NTCIP Camera Compatibility Tester shall allow the user to test if the CHART Poll Camera command operates properly	NTCIP Camera	R7VerifyNTCIPCamer aCompatibility.ucd	
SR3	MONITOR TRAFFIC AND ROADWAYS		N/A	N/A
SR3.6	UTILIZE VIDEO		N/A	N/A
SR3.6.1	The system shall allow a suitably privileged user to control cameras.		N/A	N/A



Tag	Requirement	Feature	Use Cases	Other Design Elements
SR3.6.1.3	CHART II shall establish and maintain communication with the camera for the duration of the control session.	NTCIP Camera	SendCameraCommands .ucd ManageCameraControl. .ucd	NTCIPCameraProtocolHdlr:connect.etd
SR3.6.1.3.1	An operator shall be notified if communications with the camera is lost during a camera control.	NTCIP Camera	SendCameraCommands .ucd ManageCameraControl. .ucd	
SR3.6.1.3.2	The camera control session shall have a configurable maximum no activity duration, after which the control session shall be dropped.	NTCIP Camera	SendCameraCommands .ucd ManageCameraControl. .ucd	
SR3.6.1.3.2.1	The controlling operator shall be informed if the control session has been dropped	NTCIP Camera	SendCameraCommands .ucd ManageCameraControl. .ucd	
SR3.6.1.4	A suitably privileged CHART II operator shall have the capability to initiate camera control.		N/A	N/A
SR3.6.1.4.3	A suitably privileged operator shall be able to pan or tilt a camera for which a control session is open.		N/A	N/A
SR3.6.1.4.3.5	The system shall allow a suitably privileged user to pan or tilt a COHU camera for which a control session is open.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR3.6.1.4.3.6	The system shall allow a suitably privileged user to pan or tilt an NTCIP camera for which a control session is open.	NTCIP Camera	ManageCamera.ucd SendCameraCommands .ucd	CameraControlModule.cad NTCIPCameraProtocolHdlr:adjpan.etd

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR3.6.1.4.3.6.1	The system shall automatically adjust the pan/tilt speed of an NTCIP camera based on zoom level.	NTCIP Camera	ManageCamera.ucd	CameraControlModule.cad NTCIPCameraProtocolHdlr:calculatecontorlSpeeds. etd NTCIPCameraProtocolHdlr:adjZoom.etd NTCIPCameraProtocolHdlr:getZoomPosition.etd
SR3.6.1.4.3.6.1.1	For NTCIP cameras, the system shall automatically adjust pan speed, on a linear scale, between a configurable minimum pan speed (configurable per camera) when the camera is fully zoomed in and a configurable maximum pan speed (configurable per camera) when the camera is fully zoomed out	NTCIP Camera	ManageCamera.ucd	CameraControlModule.cad NTCIPCameraProtocolHdlr:calculatecontorlSpeeds. etd NTCIPCameraProtocolHdlr:adjZoom.etd NTCIPCameraProtocolHdlr:getZoomPosition.etd
SR3.6.1.4.3.6.1.2	For NTCIP cameras, the system shall automatically adjust tilt speed, on a linear scale, between a configurable minimum tilt speed (configurable per camera) when the camera is fully zoomed in and a configurable maximum tilt speed (configurable per camera) when the camera is fully zoomed out	NTCIP Camera	ManageCamera.ucd	CameraControlModule.cad NTCIPCameraProtocolHdlr:calculatecontorlSpeeds. etd NTCIPCameraProtocolHdlr:adjZoom.etd NTCIPCameraProtocolHdlr:getZoomPosition.etd
SR3.6.1.4.4	A suitably privileged operator shall be able to zoom a camera for which a control session is open.		N/A	N/A
SR3.6.1.4.4.2	The system shall allow a suitably privileged user to zoom a COHU camera for which a control session is open.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR3.6.1.4.4.3	The system shall allow a suitably privileged user to zoom an NTCIP camera for which a control session is open.	NTCIP Camera	ManageCamera.ucd SendCameraCommands .ucd	CameraControlModule.cad NTCIPCameraProtocolHdlr:adjZoom.etd
SR3.6.1.4.5	A suitably privileged operator shall be able to focus a camera for which a control session is open.		N/A	N/A

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR3.6.1.4.5.5	The system shall allow a suitably privileged user to adjust the focus of a COHU camera for which a control session is open.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR3.6.1.4.5.6	The system shall allow a suitably privileged user to adjust the focus of an NTCIP camera for which a control session is open.	NTCIP Camera	ManageCamera.ucd SendCameraCommands .ucd	CameraControlModule.cad
SR3.6.1.4.6	A suitably privileged operator shall be able to adjust iris control of a camera for which a control session is open.		N/A	N/A
SR3.6.1.4.6.5	The system shall allow a suitably privileged user to toggle the auto iris mode of a COHU camera for which a control session is open.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR3.6.1.4.6.6	The system shall allow a suitably privileged user to adjust the iris of a COHU camera for which a control session is open.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR3.6.1.4.6.7	The system shall allow a suitably privileged user to toggle the auto iris mode of an NTCIP camera for which a control session is open.	NTCIP Camera	ManageCamera.ucd SendCameraCommands .ucd	CameraControlModule.cad
SR3.6.1.4.6.8	The system shall allow a suitably privileged user to adjust the iris of an NTCIP camera for which a control session is open.	NTCIP Camera	ManageCamera.ucd SendCameraCommands .ucd	CameraControlModule.cad

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR3.6.1.4.8	A suitably privileged operator shall be able to move a camera to a predefined preset position for which a control session is open	NTCIP Camera	DisplayCamera.ucd ManageCamera.ucd	CameraControlModule.cad NTCIPCameraProtocolHdlr:moveToPreset.etd NTCIPCameraProtocolHdlr:setPresetTitle.etd NTCIPCameraProtocolHdlr:setLabelText.etd
SR3.6.1.4.8.1	The system shall indicate the current preset number and description, if the camera is currently at a preset.	NTCIP Camera	ManageCamera.ucd	NTCIPCameraProtocolHdlr:moveToPreset.etd NTCIPCameraProtocolHdlr:setPresetTitle.etd NTCIPCameraProtocolHdlr:setLabelText.etd
SR3.6.1.4.9	The system shall allow a suitably privileged operator to maintain CCTV (camera) presets.	NTCIP Camera	ManageCamera.ucd	NTCIPCameraProtocolHdlr:storePreset.etd NTCIPCameraProtocolHdlr:setPresetTitle.etd NTCIPCameraProtocolHdlr:setLabelText.etd
SR3.6.1.4.9.4	A stored preset shall include an operator-specified title to appear on the camera image display for those camera types which support that functionality.	NTCIP Camera	ManageCamera.ucd	NTCIPCameraProtocolHdlr:storePreset.etd NTCIPCameraProtocolHdlr:setPresetTitle.etd NTCIPCameraProtocolHdlr:setLabelText.etd
SR3.6.1.4.9.4.3	Preset titles and positions shall be stored on the camera, for those COHU cameras that support such an option.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR3.6.1.4.9.4.4	Preset titles and positions shall be stored on the camera, for those NTCIP cameras that support such an option.	NTCIP Camera	ManageCamera.ucd SendCameraCommands.ucd	NTCIPCameraProtocolHdlr:storePreset.etd NTCIPCameraProtocolHdlr:setPresetTitle.etd NTCIPCameraProtocolHdlr:setLabelText.etd
SR3.6.1.4.9.5	Camera preset positions and titles shall be stored in the CHART II database.			CameraControlModule:SavePreset.etd
SR3.6.1.4.9.5.3	The number of presets that may be stored for a camera shall not exceed 10.	NTCIP Camera	ManageCamera.ucd	CameraControlModule:SavePreset.etd
SR3.6.1.4.10	A suitably privileged operator shall be able to reset a camera for which a control session is open.		N/A	N/A

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR3.6.1.4.10.2	The system shall allow a suitably privileged user to reset a COHU camera for which a control session is open.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR3.6.1.4.13	A suitably privileged operator shall be able to directly control the titles which appear on the camera image, for cameras which support direct setting of line 1 and 2 of the camera titles, provided a control session is open for that camera.		N/A	N/A
SR3.6.1.4.13.3	The system shall allow a suitably privileged user to directly control the line one title line that appears on the COHU camera image, provided that a control session is open for the camera.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR3.6.1.4.13.4	The system shall allow a suitably privileged user to directly control the line two title line that appears on the COHU camera image, provided that a control session is open for the camera.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR3.6.1.4.13.5	The system shall allow a suitably privileged user to directly control the line one title line that appears on the NTCIP camera image, provided that a control session is open for the camera.	NTCIP Camera	ManageCamera.ucd SendCameraCommands .ucd	NTCIPCameraProtocolHdlr:setLabelText.etd
SR3.6.1.4.13.6	The system shall allow a suitably privileged user to directly control the line two title line that appears on the NTCIP camera image, provided that a control session is open for the camera.	NTCIP Camera	ManageCamera.ucd SendCameraCommands .ucd	NTCIPCameraProtocolHdlr:setLabelText.etd
SR3.6.1.4.17	The system shall allow a suitably privileged user to override control of a camera that is under the control of another user.		N/A	N/A
SR3.6.1.4.17.5	The system shall allow a suitably privileged user to override control of a COHU camera that is under the control of another user.	NTCIP Camera	(Existing COHU rqmt included for reference)	

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR3.6.1.4.17.6	The system shall allow a suitably privileged user to override control of an NTCIP camera that is under the control of another user.	NTCIP Camera	ManageCamera.ucd ManageCameraControl.ucd	
SR3.6.1.5	Cameras shall be polled at a configurable interval to verify control status.	NTCIP Camera	ManageCameraControl.ucd	
SR3.6.1.5.1	Cameras that do not respond shall be identified as having communications problems.	NTCIP Camera	ManageCameraControl.ucd	
SR3.6.1.10	The system will support control of COHU MPC cameras, COHU 3955 cameras, NTCIP compatible cameras, and Surveyor VFT cameras.	NTCIP Camera	ManageCamera.ucd ManageCameraControl.ucd	RequestCameraControl.etd
SR3.6.1.11	The system shall support standards based protocols for communicating with camera control sending devices wherever possible, except when proprietary protocols are the only option for communicating with vendor devices.	NTCIP Camera	SendCameraCommands.ucd	
SR3.6.1.11.1	The system shall support camera control over an IP network.	NTCIP Camera		
SR3.6.1.11.2	The system shall support direct camera control over a COM port.		N/A	N/A
SR3.6.1.11.2.4	The system shall support direct camera control of a single NTCIP camera on a COM port.	NTCIP Camera	SendCameraCommands.ucd	PortLocatorClasses.cad DataPortUtility:receive.etd DataPortUtility:receiveFromDirectPort.etd DataPortUtility:send.etd
SR3.6.1.12	The system shall allow a user with control of a camera to release control of the camera.		N/A	N/A
SR3.6.1.12.3	The system shall allow a user with control of a COHU camera to release control of the camera.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR3.6.1.12.4	The system shall allow a user with control of an NTCIP camera to release control of the camera.	NTCIP Camera	ManageCamera.ucd ManageCameraControl.ucd	

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR3.6.1.13	The system shall automatically release control of a camera when the user closes the camera's control window.		N/A	N/A
SR3.6.1.13.2	The system shall automatically release control of a COHU camera when the user closes the camera's control window.	NTCIP Camera	(Existing COHU rqmt included for reference)	
SR3.6.1.13.3	The system shall automatically release control of an NTCIP camera when the user closes the camera's control window.	NTCIP Camera	ManageCamera.ucd ManageCameraControl.ucd	
SR4	MANAGE EVENTS		N/A	N/A
SR4.2	OPEN EVENT		N/A	N/A
SR4.2.2	RECORD EVENT DETAILS		N/A	N/A
SR4.2.2.4	CAPTURE WEATHER AND ROADWAY CONDITIONS. The system shall capture the environmental and roadway conditions for an event.	Weather Integration	Preselect Road Surface Condition, Log Weather Station Data, Display Weather Station Conditions	N/A (general requirement - see sub reqs)
SR4.2.2.4.1	The system shall capture roadway information for an event, if available.	Weather Integration	Preselect Road Surface Condition, Log Weather Station Data, Display Weather Station Conditions	N/A (general requirement - see sub reqs)
SR4.2.2.4.1.1	The system shall automatically select the road surface condition indicated by a nearby weather station for an event when the event is opened, if the event has known coordinates, and if weather data is available at the time of the opening of the event.	Weather Integration	Preselect Road Surface Condition	TrafficEventModuleClassesR7 CD, UtilityClasses3 CD, TrafficEventFactoryImpl : createTrafficEventHelper SD, TrafficEventFactoryImpl : queryRoadSurfaceConditionWeatherInfoIfApplicable SD, TrafficEventGroup : queryRoadSurfaceConditionWeatherInfo SD, TrafficEventGroup : handleRoadSurfaceConditionWeatherInfo SD, SynchAsynchQueryUtil : executeTimeLimitedQuery SD

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR4.2.2.4.1.1.1	The system shall use the nearest weather station configured with roadway sensors, with recent data, within a configurable radius from the event to select the road surface condition.	Weather Integration	Query Nearby Weather Station Data	TrafficEventGroup : queryRoadSurfaceConditionWeatherInfo SD, WeatherDataRequestHandler : handleWeatherDataRequest SD
SR4.2.2.4.1.1.1.1	The system shall allow the administrator to configure the radius for including a weather stations' data in an event.	Weather Integration	Configure Weather Settings	SystemProfileReqHdlr : getWeatherSettingsForm SD, SystemProfileReqHdlr : setWeatherSettings SD, chartlite.data.MiscDataClasses CD
SR4.2.2.4.1.1.1.2	The system shall allow the administrator to configure the maximum age for including a weather station's data in an event.	Weather Integration	Configure Weather Settings	SystemProfileReqHdlr : getWeatherSettingsForm SD, SystemProfileReqHdlr : setWeatherSettings SD, chartlite.data.MiscDataClasses CD
SR4.2.2.4.1.1.2	The system shall select the worst road surface condition from a subset of the roadway sensors reporting to the selected weather station.	Weather Integration	Query Nearby Weather Station Data	WeatherDataRequestHandler : calculateRoadSurfaceConditions SD
SR4.2.2.4.1.1.2.1	The system shall select the worst road surface condition from among the station's roadway sensors located on the event's primary route and direction, if any.	Weather Integration	Query Nearby Weather Station Data	WeatherDataRequestHandler : calculateRoadSurfaceConditions SD
SR4.2.2.4.1.1.2.2	The system shall select the worst road surface condition from among the station's roadway sensors located on the event's primary route if any, if no roadway sensors are found matching the event's primary route and direction.	Weather Integration	Query Nearby Weather Station Data	WeatherDataRequestHandler : calculateRoadSurfaceConditions SD
SR4.2.2.4.1.1.2.4	The system shall select the worst road surface condition from among all of the weather station's roadway sensors if no roadway sensors are found matching the event's primary route.	Weather Integration	Query Nearby Weather Station Data	WeatherDataRequestHandler : calculateRoadSurfaceConditions SD
SR4.2.2.4.1.1.3	DISPLAY WEATHER STATION CONDITIONS The system shall display weather station information to the user captured at the time the system selected the road surface condition but only if the system selected a road surface condition.	Weather Integration	Display Weather Station Conditions	TrafficEventGroup : handleRoadSurfaceConditionWeatherInfo; Prototype



Tag	Requirement	Feature	Use Cases	Other Design Elements
SR4.2.2.4.1.1.3.1	The system shall display the name of the weather system from which the data was obtained.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.2	The system shall display a description of the location of the weather station.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.3	The system shall display the distance from the traffic event to the weather station.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.4	The system shall display the road surface condition reported by the weather station.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.5	The system shall display the wind speed reported by the weather station, if available.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.6	The system shall display the wind direction reported by the weather station, if available.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.7	The system shall display the visibility reported by the weather station, if available.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.8	The system shall display the air temperature reported by the weather station, if available.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.9	The system shall display the precipitation type reported by the weather station, if available.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.10	The system shall display the precipitation intensity reported by the weather station, if available.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.11	The system shall display the time that the weather data was collected from the sensors.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.12	The system shall display data for each roadway sensor managed by the weather station.	Weather Integrati on	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.12.1	The system shall display a description of the location of the roadway sensor.	Weather Integrati	Display Weather Station Conditions	Prototype / JAD only

Tag	Requirement	Feature	Use Cases	Other Design Elements
		on		
SR4.2.2.4.1.1.3.12.2	The system shall display the road surface temperature reported by the roadway sensor, if available.	Weather Integration	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.1.3.12.3	The system shall display the road surface condition reported by the roadway sensor, if available.	Weather Integration	Display Weather Station Conditions	Prototype / JAD only
SR4.2.2.4.1.2	The system shall allow the user to invoke the Intranet Map page to view nearby weather stations, if the Intranet Map is accessible from the user's browser.	Weather Integration	View Traffic Events	Prototype / JAD, ServletBaseClasses CD
SR4.2.2.4.1.3	The system shall add entries to the traffic event history log summarizing the system-selected weather station and roadway sensor information.	Weather Integration	Log Weather Station Data	TrafficEventGroup : handleRoadSurfaceConditionWeatherInfo SD
SR4.2.2.4.1.3.1	The system shall log the same weather data fields displayed to the user, as defined in the Display Weather Station Conditions requirements (SR.4.2.2.4.1.1.3 and its subrequirements).	Weather Integration	Log Weather Station Data	N/A (Use Case Only)
SR4.2.2.4.1.3.2	The system shall add an entry to the traffic event log whenever the road surface condition is selected by the system.	Weather Integration	Preselect Road Surface Condition, Log Weather Station Data	TrafficEventGroup : handleRoadSurfaceConditionWeatherInfo SD
SR4.2.2.4.1.3.3	The system shall add a weather entry to the traffic event log when the event is closed, if weather information is available.	Weather Integration	Close Event, Log Weather Station Data	TrafficEventGroup : close SD, TrafficEventGroup : handleRoadSurfaceCondition SD
SR4.2.2.4.1.4	The system shall allow the user to invoke the details page for the selected weather station within the SCAN Web user interface, if the SCAN Web user interface is accessible from the user's browser.	Weather Integration	Display Weather Station Conditions	Prototype / JAD, ServletBaseClasses CD

Tag	Requirement	Feature	Use Cases	Other Design Elements
SR10	SYSTEM INTEGRATION		N/A	N/A
SR10.10	The system shall integrate TSS data from external systems together with internally created CHART TSS data.		N/A	N/A
SR10.10.2	The system shall allow a suitably privileged user to manage the importing of a TSS from an external system into the CHART system.		N/A	N/A
SR10.10.2.3	The system shall allow a suitably privileged user to delete an external TSS from the CHART system. (Note: This does not prevent its use as candidate external TSS.)		N/A	N/A
SR10.10.2.3.2	The external TSS candidates list shall include a notice that setting any TSS from "included" state to a state other than "included" will result in that external TSS being deleted from the system along with any location or bearing attributes it may have been given.	Map		

The following table shows how the requirements in the Mapping R6 Requirements document map to design elements contained in this design.

Tag	Text	Feature	Use Cases	Other Design Elements
SR6	Detailed Map Layer Requirements		N/A	N/A
SR6.4	Traffic Speed Sensor (TSS/RTMS)		N/A	N/A
SR6.4.3	Detector arrow shall be rotated according to the bearing generated by the CHART system.	Map	View TSSs on Intranet & Internet map	DataSynchronization: HumanMachine

Tag	Text	Feature	Use Cases	Other Design Elements
SR6.4.6	When user hovers over the detector icon, tool tip shall display following information: Detector location, Last data report date and time, Zone Group direction(s), Indication of Speed and Owning organization.	ExternalTSS	View TSSs on Intranet & Internet map	Configuration - Intranet Map (Class Diagram)
SR6.4.6.1	The system shall display the average speed value for each displayable detector zone group if the current user has CHART's ViewVSODetailedData right for the detector's owning organization.	ExternalTSS, Map	View TSSs on Intranet & Internet map	Configuration - Intranet Map (Class Diagram)
SR6.4.9	Zone Group Display	DataSync		
SR6.4.9.1	The system shall display the zone group name as specified by the CHART system.	DataSync	View Zone Group Display on TSS	CHART Data Exporter : CHARTMap.Handlers.TssInventoryHandler : CD
SR6.4.9.2	The system shall display a detector zone group only if the 'Display On Maps' indicator from the CHART system indicates to do so.	DataSync	View Zone Group Display on TSS	CHART Data Exporter : CHARTMap.Handlers.TssInventoryHandler : CD
SR6.4.9.2.1	If a TSS is changed to have no displayable zone groups (in CHART), it will not be displayed on the map.	Map	View Zone Group Display on TSS	CHART Data Exporter : CHARTMap.Handlers.TssInventoryHandler : CD
SR6.4.9.2.2	If a TSS has one or more displayable zone groups marked as map displayable (in CHART) the TSS will be displayed on the map.	Map	View Zone Group Display on TSS	CHART Data Exporter : CHARTMap.Handlers.TssInventoryHandler : CD
SR11	Data Exporter Synchronization Requirements	DataSync	N/A	N/A

Tag	Text	Feature	Use Cases	Other Design Elements
SR11.1	General	DataSync	N/A	N/A
SR11.1.7	The system shall maintain latitude and longitude information for SHAZAM, DMS, TSS, HAR, CAMERA, DETECTOR and Traffic Events.	DataSync		CHART Data Exporter Synchronization SD DataSynchronization: HumanMachine
SR11.2	Synchronize Add Event	DataSync	Synchronize Add Events	CHART Data Exporter Synchronization SD
SR11.2.2	The synchronization application shall add an entry to the spatial table if a new CHART Device (DMS, HAR, SHAZAM, CAMERA, DETECTOR) is found.	DataSync, DataSyncCamera	Synchronize Add Events	CHART Data Exporter Synchronization SD
SR11.3	Synchronize Update Event	DataSync	Synchronize Update Events	CHART Data Exporter Synchronization:UpdateInventory() SD
SR11.3.2	The synchronization application shall update an entry based on the unique ID in the relevant spatial table if a CHART Device (DMS, HAR, SHAZAM, CAMERA, DETECTOR) changes location.	DataSync, DataSyncCamera	Synchronize Update Events	CHART Data Exporter Synchronization:UpdateInventory() SD

Tag	Text	Feature	Use Cases	Other Design Elements
SR11.5	Functionalities retired from Device Editor	DataSync	Synchronize Add Events , Synchronize Update Events , Synchronize Remove Events	N/A
SR11.5.1	The Device Editor shall prohibit the display of CHART Devices (DMS, HAR, SHAZAM, CAMERA and DETECTOR).	DataSync, DataSyncCamera	Synchronize Add Events , Synchronize Update Events , Synchronize Remove Events	DataSynchronization: HumanMachine
SR11.5.2	The Device Editor shall not allow users to add CHART Devices (DMS, HAR, SHAZAM, CAMERA and DETECTOR) to the map. (This will be done via CHART only.)	DataSync, DataSyncCamera	Synchronize Add Events , Synchronize Update Events , Synchronize Remove Events	DataSynchronization: HumanMachine
SR11.5.3	The Device Editor shall not allow users to update CHART Devices (DMS, HAR, SHAZAM, CAMERA and DETECTOR). (This will be done via CHART only.)	DataSync, DataSyncCamera	Synchronize Add Events , Synchronize Update Events , Synchronize Remove Events	DataSynchronization: HumanMachine
SR11.5.4	The Device Editor shall not allow users to remove CHART Devices (DMS, HAR, SHAZAM, CAMERA, and DETECTOR) from the map. (This will be done via CHART only.)	DataSync, DataSyncCamera	Synchronize Add Events , Synchronize Update Events , Synchronize Remove Events	DataSynchronization: HumanMachine

Tag	Text	Feature	Use Cases	Other Design Elements
SR11.2	Synchronize Add Event	DataSync	Synchronize Add Events	CHART Data Exporter Synchronization SD
SR11.2.2	The synchronization application shall add an entry to the spatial table if a new CHART Device (DMS, HAR, SHAZAM, CAMERA, DETECTOR) is found.	DataSync, DataSyncCamera	Synchronize Add Events	CHART Data Exporter Synchronization SD
SR11.3	Synchronize Update Event	DataSync	Synchronize Update Events	CHART Data Exporter Synchronization:UpdateInventory() SD
SR11.3.2	The synchronization application shall update an entry based on the unique ID in the relevant spatial table if a CHART Device (DMS, HAR, SHAZAM, CAMERA, DETECTOR) changes location.	DataSync, DataSyncCamera	Synchronize Update Events	CHART Data Exporter Synchronization:UpdateInventory() SD
SR11.5	Functionalities retired from Device Editor	DataSync	Synchronize Add Events , Synchronize Update Events , Synchronize Remove Events	N/A
SR11.5.1	The Device Editor shall prohibit the display of CHART Devices (DMS, HAR, SHAZAM, CAMERA and DETECTOR).	DataSync, DataSyncCamera	Synchronize Add Events , Synchronize Update Events , Synchronize Remove Events	DataSynchronization: HumanMachine
SR11.5.2	The Device Editor shall not allow users to add CHART Devices (DMS,	DataSync, DataSyncCam	Synchronize Add Events ,	DataSynchronization: HumanMachine

Tag	Text	Feature	Use Cases	Other Design Elements
	HAR, SHAZAM, CAMERA and DETECTOR) to the map. (This will be done via CHART only.)	era	Synchronize Update Events , Synchronize Remove Events	
SR11.5.3	The Device Editor shall not allow users to update CHART Devices (DMS, HAR, SHAZAM, CAMERA and DETECTOR). (This will be done via CHART only.)	DataSync, DataSyncCam era	Synchronize Add Events , Synchronize Update Events , Synchronize Remove Events	DataSynchronization: HumanMachine
SR11.5.4	The Device Editor shall not allow users to remove CHART Devices (DMS, HAR, SHAZAM, CAMERA, and DETECTOR) from the map. (This will be done via CHART only.)	DataSync, DataSyncCam era	Synchronize Add Events , Synchronize Update Events , Synchronize Remove Events	DataSynchronization: HumanMachine
SR6	Detailed Map Layer Requirements		N/A	N/A
SR6.4	Traffic Speed Sensor (TSS/RTMS)			



Tag	Text	Feature	Use Cases	Other Design Elements
SR6.4.3	Detector arrow shall be rotated according to the bearing generated by the CHART system.	Map	View TSSs on Intranet & Internet map	DataSynchronization: HumanMachine
SR6.4.6	When user hovers over the detector icon, tool tip shall display following information: Detector location, Last data report date and time, Zone Group direction(s), Indication of Speed and Owning organization.	ExternalTSS	View TSSs on Intranet & Internet map	Configuration - Intranet Map (Class Diagram)
SR6.4.6.1	The system shall display the average speed value for each displayable detector zone group if the current user has CHART's ViewVSODetailedData right for the detector's owning organization.	ExternalTSS, Map	View TSSs on Intranet & Internet map	Configuration - Intranet Map (Class Diagram)
SR6.4.9	Zone Group Display	DataSync		
SR6.4.9.1	The system shall display the zone group name as specified by the CHART system.	DataSync	View Zone Group Display on TSS	CHART Data Exporter : CHARTMap.Handlers.TssInventoryHandler : CD
SR6.4.9.2	The system shall display a detector zone group only if the 'Display On Maps' indicator from the CHART system indicates to do so.	DataSync	View Zone Group Display on TSS	CHART Data Exporter : CHARTMap.Handlers.TssInventoryHandler : CD
SR6.4.9.2.1	If a TSS is changed to have no displayable zone groups (in CHART), it will not be displayed on the map.	Map	View Zone Group Display on TSS	CHART Data Exporter : CHARTMap.Handlers.TssInventoryHandler : CD

Tag	Text	Feature	Use Cases	Other Design Elements
SR6.4.9.2.2	If a TSS has one or more displayable zone groups marked as map displayable (in CHART) the TSS will be displayed on the map.	Map	View Zone Group Display on TSS	CHART Data Exporter : CHARTMap.Handlers.TssInventoryHandler : CD
SR11	Data Exporter Synchronization Requirements	DataSync	N/A	N/A
SR11.1	General	DataSync	N/A	N/A
SR11.1.7	The system shall maintain latitude and longitude information for SHAZAM, DMS, TSS, HAR, CAMERA, DETECTOR and Traffic Events.	DataSync		CHART Data Exporter Synchronization SD DataSynchronization: HumanMachine
SR11.2	Synchronize Add Event	DataSync	Synchronize Add Events	CHART Data Exporter Synchronization SD
SR11.2.2	The synchronization application shall add an entry to the spatial table if a new CHART Device (DMS, HAR, SHAZAM, CAMERA, DETECTOR) is found.	DataSync, DataSyncCam era	Synchronize Add Events	CHART Data Exporter Synchronization SD
SR11.3	Synchronize Update Event	DataSync	Synchronize Update Events	CHART Data Exporter Synchronization:UpdateInventory() SD

## 12Acronyms/Glossary

<b>GIS</b>	<b>Geographic Information System</b> (GIS) is any system that captures, stores, analyzes, manages, and presents data that are linked to location
<b>Home Page Map</b>	The map component shown on the home page of the CHART user interface.
<b>Integrated Map</b>	The mapping components that are being built into the CHART user interface as part of Release 6 of the CHART application.
<b>Intranet Map</b>	The CHART Mapping application that is not integrated into the CHART user interface.
<b>Location Alias</b>	A pre-defined location (lat/lon) that has been stored with some name attributes to allow operators to utilize the location repeatedly.
<b>Maintenance Portal</b>	A customized version of the CHART GUI tailored to device maintenance personnel.
<b>Nearby Devices Map</b>	Map shown on the details page for a traffic event that shows only the target traffic event and the devices that are near it.
<b>NTCIP</b>	National Transportation Communications for ITS Protocol. A family of standards designed to achieve interoperability and interchangeability between computers and electronic traffic control equipment from different manufacturers.
<b>Object Location Map</b>	Map component that is used in conjunction with the object location form when setting the location of a traffic event or device.
<b>Open Layers</b>	Open source JavaScript mapping API utilized by the integrated map components in the CHART GUI.
<b>REST</b>	Representational State Transfer - a web services architecture style used in CHART that leverages web technologies such as http and XML
<b>RWIS</b>	Roadway Weather Information System
<b>Standard GUI</b>	The CHART GUI when <i>not</i> accessed via the maintenance portal.
<b>TSS</b>	<b>Transportation Sensor System</b>
<b>WMS</b>	A Web Map Service (WMS) is a standard protocol for serving georeferenced map images over the Internet that are generated by a map server using data from a GIS database.
<b>Wx</b>	Weather Station